

Oracle® Application Server

PL/SQL Web Toolkit Reference

Release 4.0.8.1

September 1999

Part No. A60123-03

ORACLE®

Oracle Application Server Release 4.0.8.1 PL/SQL Web Toolkit Reference

Part No. A60123-03

Copyright © 1996, 1999, Oracle Corporation. All rights reserved.

Primary Author: Alka Srivastava

Contributors: Sanjay Patil, Sanjay Singh

The programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the programs.

The programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the programs is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. Except as may be expressly permitted in your license agreement for these programs, no part of these programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

Oracle is a registered trademark, and the Oracle logo, NLS*WorkBench, Pro*COBOL, Pro*FORTRAN, Pro*Pascal, SQL*Loader, SQL*Module, SQL*Net, SQL*Plus, Oracle7, Oracle Server, Oracle Server Manager, Oracle Call Interface, Oracle7 Enterprise Backup Utility, Oracle TRACE, Oracle WebServer, Oracle Web Application Server, Oracle Application Server, Oracle Network Manager, Secure Network Services, Oracle Parallel Server, Advanced Replication Option, Oracle Data Query, Cooperative Server Technology, Oracle Toolkit, Oracle MultiProtocol Interchange, Oracle Names, Oracle Book, Pro*C, and PL/SQL are trademarks or registered trademarks of Oracle Corporation. All other company or product names mentioned are used for identification purposes only and may be trademarks of their respective owners.

Contents

Preface.....	xi
--------------	----

1 The htp and htf Packages

Summary.....	1-1
htp.address	1-6
htp.anchor, htp.anchor2	1-7
htp.appletopen, htp.appletclose.....	1-9
htp.area.....	1-11
htp.base.....	1-12
htp.basefont	1-13
htp.bgsound	1-14
htp.big	1-15
htp.blockquoteOpen, htp.blockquoteClose	1-16
htp.bodyOpen, htp.bodyClose	1-17
htp.bold.....	1-18
htp.center.....	1-19
htp.centerOpen, htp.centerClose	1-20
htp.cite.....	1-21
htp.code.....	1-22
htp.comment	1-23
htp.dfn.....	1-24
htp.dirlistOpen, htp.dirlistClose	1-25
htp.div	1-26
htp.dlistOpen, htp.dlistClose	1-27
htp.dlistDef	1-28

htp.dlistTerm	1-29
htp.download_file	1-30
htp.get_download_files_list	1-31
htp.emphasis, htp.em	1-32
htf.escape_sc	1-33
htf.escape_url	1-34
htp.fontOpen, htp.fontClose	1-35
htf.format_cell	1-36
htp.formCheckbox	1-37
htp.formOpen, htp.formClose	1-38
htp.formHidden	1-39
htp.formImage	1-40
htp.formPassword	1-41
htp.formRadio	1-42
htp.formReset	1-43
htp.formSelectOpen, htp.formSelectClose	1-44
htp.formSelectOption	1-46
htp.formSubmit	1-47
htp.formText	1-48
htp.formTextarea, htp.formTextarea2	1-49
htp.formTextareaOpen, htp.formTextareaOpen2, htp.formTextareaClose	1-51
htp.frame	1-53
htp.framesetOpen, htp.framesetClose	1-54
htp.headOpen, htp.headClose	1-55
htp.header	1-56
htp.htmlOpen, htp.htmlClose	1-57
htp.img, htp.img2	1-58
htp.isindex	1-60
htp.italic	1-61
htp.keyboard, htp.kbd	1-62
htp.line, htp.hr	1-63
htp.linkRel	1-64
htp.linkRev	1-65
htp.listHeader	1-66
htp.listingOpen, htp.listingClose	1-67

htp.listItem	1-68
htp.mailto	1-69
htp.mapOpen , htp.mapClose	1-70
htp.menuListOpen , htp.menuListClose	1-71
htp.meta	1-72
htp.nl , htp.br	1-73
htp.nobr	1-74
htp.noFramesOpen , htp.noFramesClose	1-75
htp.olistOpen , htp.olistClose	1-76
htp.para , htp.paragraph	1-77
htp.param	1-78
htp.plaintext	1-79
htp.preOpen , htp.preClose	1-80
htp.print , htp.prn	1-81
htp.prints , htp.ps	1-82
htp.s	1-83
htp.sample	1-84
htp.script	1-85
htp.small	1-86
htp.strike	1-87
htp.strong	1-88
htp.style	1-89
htp.sub	1-90
htp.sup	1-91
htp.tableCaption	1-92
htp.tableData	1-93
htp.tableHeader	1-94
htp.tableOpen , htp.tableClose	1-95
htp.tableRowOpen , htp.tableRowClose	1-96
htp.teletype	1-97
htp.title	1-98
htp.ulistOpen , htp.ulistClose	1-99
htp.underline	1-100
htp.variable	1-101
htp.wbr	1-102

2 The owa_content Package

Summary	2-1
owa_content.bigstring_arr data type	2-3
owa_content.delete_attribute procedure	2-4
owa_content.delete_document procedure	2-5
owa_content.delete_documents procedure	2-6
owa_content.document_exists function	2-7
owa_content.get_attribute procedure	2-8
owa_content.get_attributes function	2-11
owa_content.get_author function	2-12
owa_content.get_char_set function	2-13
owa_content.get_content_type function	2-14
owa_content.get_description function	2-15
owa_content.get_encoding function	2-16
owa_content.get_expiration function	2-17
owa_content.get_language function	2-18
owa_content.get_length function	2-19
owa_content.get_title function	2-20
owa_content.list_attributes procedure	2-21
owa_content.list_documents function	2-22
owa_content.list_system_attributes procedure	2-23
owa_content.list_user_attributes procedure	2-24
owa_content.number_arr data type	2-25
owa_content.rename_document procedure	2-26
owa_content.set_attribute procedure	2-27
owa_content.set_author procedure	2-28
owa_content.set_char_set procedure	2-29
owa_content.set_content_type procedure	2-30
owa_content.set_description procedure	2-31
owa_content.set_encoding procedure	2-32
owa_content.set_expiration procedure	2-33
owa_content.set_language procedure	2-34
owa_content.set_title procedure	2-35
owa_content.string_arr data type	2-36

3 The owa_cookie Package

Summary	3-1
owa_cookie.cookie data type	3-2
owa_cookie.get function	3-3
owa_cookie.get_all procedure	3-4
owa_cookie.remove procedure	3-5
owa_cookie.send procedure	3-6

4 The owa_image Package

Summary	4-1
owa_image.NULL_POINT package variable	4-2
owa_image.point data type	4-3
owa_image.get_x function	4-4
owa_image.get_y function	4-5

5 The owa_opt_lock Package

Summary	5-1
owa_opt_lock.vcArray data type	5-2
owa_opt_lock.checksum function	5-3
owa_opt_lock.get_rowid function	5-4
owa_opt_lock.store_values procedure	5-5
owa_opt_lock.verify_values function	5-6

6 The owa_pattern Package

Summary	6-1
Regular Expressions	6-2
Wildcard Tokens.....	6-2
Quantifiers	6-3
Flags.....	6-3
owa_pattern.amatch function	6-4
owa_pattern.change function and procedure	6-6
owa_pattern.getpat procedure	6-8
owa_pattern.match function	6-9
owa_pattern.pattern data type	6-11

7 The owa_sec Package

Summary	7-1
owa_sec.get_client_hostname function	7-2
owa_sec.get_client_ip function	7-3
owa_sec.get_password function	7-4
owa_sec.get_user_id function	7-5
owa_sec.set_authorization procedure	7-6
owa_sec.set_protection_realm procedure	7-8

8 The owa_text Package

Summary	8-1
owa_text.add2multi procedure	8-2
owa_text.multi_line data type	8-3
owa_text.new_row_list	8-4
owa_text.print_multi procedure	8-5
owa_text.print_row_list procedure	8-6
owa_text.row_list data type	8-7
owa_text.stream2multi procedure	8-8
owa_text.vc_arr data type	8-9

9 The owa_util Package

Summary	9-1
owa_util.bind_variables function	9-3
owa_util.calendarprint procedure	9-4
owa_util.cellsprint procedure	9-6
owa_util.choose_date procedure	9-9
owa_util.dateType data type	9-10
owa_util.get_cgi_env function	9-11
owa_util.get_owa_service_path function	9-12
owa_util.get_procedure function	9-13
owa_util.http_header_close procedure	9-14
owa_util.ident_arr data type	9-15
owa_util.ip_address data type	9-16
owa_util.listprint procedure	9-17

owa_util.mime_header procedure	9-19
owa_util.print_cgi_env procedure.....	9-20
owa_util.redirect_url procedure.....	9-21
owa_util.showpage procedure	9-22
owa_util.showsource procedure	9-23
owa_util.signature procedure.....	9-24
owa_util.status_line procedure.....	9-25
owa_util.tablePrint function.....	9-26
owa_util.todate function	9-30
owa_util.who_called_me procedure	9-31

Index

Preface

Audience

This guide is a reference for the PL/SQL Web Toolkit. The intended audience for this guide are people who develop PL/SQL applications using the Toolkit **The Oracle Application Server Documentation Set**.

This table lists the Oracle Application Server documentation set.

Title of Book	Part No.
Oracle Application Server 4.0.8 Documentation Set	A66971-03
Oracle Application Server Overview and Glossary	A60115-03
Oracle Application Server Installation Guide for Sun SPARC Solaris 2.x	A58755-03
Oracle Application Server Installation Guide for Windows NT	A58756-03
Oracle Application Server Administration Guide	A60172-03
Oracle Application Server Security Guide	A60116-03
Oracle Application Server Performance and Tuning Guide	A60120-03
Oracle Application Server Developer's Guide: PL/SQL and ODBC Applications	A66958-02
Oracle Application Server Developer's Guide: JServlet Applications	A73043-01
Oracle Application Server Developer's Guide: LiveHTML and Perl Applications	A66960-02
Oracle Application Server Developer's Guide: EJB, ECO/Java and CORBA Applications	A69966-01
Oracle Application Server Developer's Guide: C++ CORBA Applications	A70039-01
Oracle Application Server PL/SQL Web Toolkit Reference	A60123-03

Title of Book	Part No.
Oracle Application Server PL/SQL Web Toolkit Quick Reference	A60119-03
Oracle Application Server JServlet Toolkit Reference	A73045-01
Oracle Application Server JServlet Toolkit Quick Reference	A73044-01
Oracle Application Server Cartridge Management Framework	A58703-03
Oracle Application Server 4.0.8.1 Release Notes	A66106-04

Conventions

This table lists the typographical conventions used in this manual.

Convention	Example	Explanation
bold	oas.h owsctl wrbcfg www.oracle.com	Identifies file names, utilities, processes, and URLs
italics	<i>file1</i>	Identifies a variable in text; replace this placeholder with a specific value or string.
angle brackets	<filename>	Identifies a variable in code; replace this placeholder with a specific value or string.
courier	owsctl start wrb	Text to be entered exactly as it appears. Also used for functions.
square brackets	[-c string] [on off]	Identifies an optional item. Identifies a choice of optional items, each separated by a vertical bar (), any one option can be specified.
braces	{yes no}	Identifies a choice of mandatory items, each separated by a vertical bar ().
ellipses	n,...	Indicates that the preceding item can be repeated any number of times.

The term “Oracle Server” refers to the database server product from Oracle Corporation.

The term “**oracle**” refers to an executable or account by that name.

The term “*oracle*” refers to the owner of the Oracle software.

Technical Support Information

Oracle Global Support can be reached at the following numbers:

- In the USA: **Telephone: 1.650.506.1500**
- In Europe: **Telephone: +44 1344 860160**
- In Asia-Pacific: **Telephone: +61. 3 9246 0400**

Please prepare the following information before you call, using this page as a checklist:

- your CSI number (if applicable) or full contact details, including any special project information
- the complete release numbers of the Oracle Application Server and associated products
- the operating system name and version number
- details of error codes and numbers and descriptions. Please write these down as they occur. They are critical in helping WWCS to quickly resolve your problem.
- a full description of the issue, including:
 - **What** - What happened? For example, the command used and its result.
 - **When** - When did it happen? For example, during peak system load, or after a certain command, or after an operating system upgrade.
 - **Where** - Where did it happen? For example, on a particular system or within a certain procedure or table.
 - **Extent** - What is the extent of the problem? For example, production system unavailable, or moderate impact but increasing with time, or minimal impact and stable.
- Keep copies of any trace files, core dumps, and redo log files recorded at or near the time of the incident. WWCS may need these to further investigate your problem. For a list of trace and log files, see "Configuration and Log Files" in the *Administration Guide*.

For installation-related problems, please have the following additional information available:

- listings of the contents of \$ORACLE_HOME (Unix) or %ORACLE_HOME% (NT) and any staging area, if used.

- installation logs (`install.log`, `sql.log`, `make.log`, and `os.log`) typically stored in the `$ORACLE_HOME/orainst` (Unix) or `%ORACLE_HOME%\orainst` (NT) directory.

Documentation Sales and Client Relations

In the United States:

- To order hardcopy documentation, call Documentation Sales: **1.800.252.0303**.
- For shipping inquiries, product exchanges, or returns, call Client Relations: **1.650.506.1500**.

In the United Kingdom:

- To order hardcopy documentation, call Oracle Direct Response:
+44 990 332200.
- For shipping inquiries and upgrade requests, call Customer Relations:
+44 990 622300.

Reader's Comment Form

Oracle Application Server PL/SQL Web Toolkit Reference

Part No. A60123-03

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have suggestions for improvement, please indicate the topic, chapter, and page number below:

Please send your comments to:

Oracle Application Server Documentation Manager
Oracle Corporation
500 Oracle Parkway
Redwood Shores, CA 94065

If you would like a reply, please provide your name, address, and telephone number below:

Thank you for helping us improve our documentation.

The htp and htf Packages

The htp and htf (hypertext procedures and hypertext functions) packages generate HTML tags that you can use to create dynamic Web pages.

For every htp procedure that generates HTML tags, there is a corresponding htf function with identical parameters. The difference is that the function passes its output to its caller and is typically used for nesting within procedures or other functions.

To look up htf functions, see the entry for the corresponding htp procedures. The string listed under “Generates” is the return value of the function.

To get information on HTML, see:

- <http://www.w3.org/pub/WWW/MarkUp/Wilbur/features.html>

For information on the Web in general, see:

- <http://www.boutell.com/faq>

Summary

HTML, HEAD, and BODY Tags

[htp.htmlOpen](#), [htp.htmlClose](#) - generate <HTML> and </HTML>

[htp.headOpen](#), [htp.headClose](#) - generate <HEAD> and </HEAD>

[htp.bodyOpen](#), [htp.bodyClose](#) - generate <BODY> and </BODY>

Comment Tag

[htp.comment](#) - generates <!-- and -->

Tags in the <HEAD> Area

`htp.base` - generates <BASE>

`htp.linkRel` - generates <LINK> with the REL attribute

`htp.linkRev` - generates <LINK> with the REV attribute

`htp.title` - generates <TITLE>

`htp.meta` - generates <META>

`htp.script` - generates <SCRIPT>

`htp.style` - generates <STYLE>

`htp.isindex` - generates <ISINDEX>

Applet Tags

`htp.appletOpen`, `htp.appletClose` - generate <APPLET> and </APPLET>

`htp.param` - generates <PARAM>

List Tags

`htp.olistOpen`, `htp.olistClose` - generate and

`htp.ulistOpen`, `htp.ulistClose` - generate and

`htp.dlistOpen`, `htp.dlistClose` - generate <DL> and </DL>

`htp.dlistTerm` - generates <DT>

`htp.dlistDef` - generates <DD>

`htp.dirlistOpen`, `htp.dirlistClose` - generate <DIR> and </DIR>

`htp.listHeader` - generates <LH>

`htp.listingOpen`, `htp.listingClose` - generate <LISTING> and </LISTING>

`htp.menuListOpen`, `htp.menuListClose` - generate <MENU> and </MENU>

`htp.listItem` - generates

Form Tags

`htp.formOpen`, `htp.formClose` - generate <FORM> and </FORM>

`htp.formCheckbox` - generates <INPUT TYPE="CHECKBOX">

`htp.formHidden` - generates <INPUT TYPE="HIDDEN">

`htp.formImage` - generates <INPUT TYPE="IMAGE">
`htp.formPassword` - generates <INPUT TYPE="PASSWORD">
`htp.formRadio` - generates <INPUT TYPE="RADIO">
`htp.formSelectOpen`, `htp.formSelectClose` - generate <SELECT> and </SELECT>
`htp.formSelectOption` - generates <OPTION>
`htp.formText` - generates <INPUT TYPE="TEXT">
`htp.formTextarea`, `htp.formTextarea2` - generate <TEXTAREA>
`htp.formTextareaOpen`, `htp.formTextareaOpen2`, `htp.formTextareaClose` - generate <TEXTAREA> and </TEXTAREA>
`htp.formReset` - generates <INPUT TYPE="RESET">
`htp.formSubmit` - generates <INPUT TYPE="SUBMIT">

Table Tags

`htp.tableOpen`, `htp.tableClose` - generate <TABLE> and </TABLE>
`htp.tableCaption` - generates <CAPTION>
`htp.tableRowOpen`, `htp.tableRowClose` - generate <TR> and </TR>
`htp.tableHeader` - generates <TH>
`htp.tableData` - generates <TD>
`htf.format_cell` - generates <TD>

IMG, HR, and A Tags

`htp.line`, `htp.hr` - generate <HR>
`htp.img`, `htp.img2` - generate
`htp.anchor`, `htp.anchor2` - generates <A>
`htp.mapOpen`, `htp.mapClose` - generate <MAP> and </MAP>

Paragraph Formatting Tags

`htp.header` - generates heading tags (<H1> to <H6>)
`htp.para`, `htp.paragraph` - generate <P>
`htp.print`, `htp.prn` - generate any text that is passed in

`ht.prints`, `ht.ps` - generate any text that is passed in; special characters in HTML are escaped

`ht.preOpen`, `ht.preClose` - generate `<PRE>` and `</PRE>`

`ht.blockquoteOpen`, `ht.blockquoteClose` - generate `<BLOCKQUOTE>` and `</BLOCKQUOTE>`

`ht.div` - generates `<DIV>`

`ht.nl`, `ht.br` - generate `
`

`ht.nobr` - generates `<NOBR>`

`ht.wbr` - generates `<WBR>`

`ht.plaintext` - generates `<PLAINTEXT>`

`ht.address` - generates `<ADDRESS>`

`ht.mailto` - generates `<A>` with the MAILTO attribute

`ht.area` - generates `<AREA>`

`ht.bgsound` - generates `<BGSOUND>`

Character Formatting Tags

`ht.basefont` - generates `<BASEFONT>`

`ht.big` - generates `<BIG>`

`ht.bold` - generates ``

`ht.center` - generates `<CENTER>` and `</CENTER>`

`ht.centerOpen`, `ht.centerClose` - generate `<CENTER>` and `</CENTER>`

`ht.cite` - generates `<CITE>`

`ht.code` - generates `<CODE>`

`ht.dfn` - generates `<DFN>`

`ht.get_download_files_list` - generate ``

`ht.fontOpen`, `ht.fontClose` - generate `` and ``

`ht.italic` - generates `<I>`

`ht.keyboard`, `ht.kbd` - generate `<KBD>` and `</KBD>`

`ht.s` - generates `<S>`

`htp.sample` - generates <SAMP>
`htp.small` - generates <SMALL>
`htp.strike` - generates <STRIKE>
`htp.strong` - generates
`htp.sub` - generates <SUB>
`htp.sup` - generates <SUP>
`htp.teletype` - generates <TT>
`htp.underline` - generates <U>
`htp.variable` - generates <VAR>

Frame Tags

`htp.frame` - generates <FRAME>
`htp.framesetOpen`, `htp.framesetClose` - generate <FRAMESET> and </FRAMESET>
`htp.noframesOpen`, `htp.noframesClose` - generate <NOFRAMES> and </NOFRAMES>

http.address

Syntax

```
http.address (
    cvalue      in      varchar2
    cnowrap    in      varchar2  DEFAULT NULL
    cclear      in      varchar2  DEFAULT NULL
    cattributes  in      varchar2  DEFAULT NULL);
htf.address (cvalue, cnowrap, cclear, cattributes) return varchar2;
```

Purpose

Generates the <ADDRESS> and </ADDRESS> tags, which specify the address, author and signature of a document.

Parameters

cvalue - the string that goes between the <ADDRESS> and </ADDRESS> tags.
cnowrap - if the value for this parameter is not NULL, the NOWRAP attribute is included in the tag.
cclear - the value for the CLEAR attribute.
cattributes - other attributes to be included as-is in the tag.

Generates>Returns

<ADDRESS CLEAR="*cclear*" NOWRAP *cattributes*>*cvalue*</ADDRESS>

http.anchor, http.anchor2

Syntax

```
http.anchor (
    curl          in      varchar2
    ctext         in      varchar2
    cname         in      varchar2  DEFAULT NULL
    cattributes   in      varchar2  DEFAULT NULL);
htf.anchor (curl, ctext, cname, cattributes) return varchar2;

http.anchor2 (
    curl          in      varchar2
    ctext         in      varchar2
    cname         in      varchar2  DEFAULT NULL
    cttarget      in      varchar2  DEFAULT NULL
    cattributes   in      varchar2  DEFAULT NULL);
htf.anchor2 (curl, ctext, cname, cttarget, cattributes) return varchar2;
```

Purpose

Generates the <A> and HTML tags, which specify the source or destination of a hypertext link. This tag accepts several attributes, but either HREF or NAME is required. HREF specifies to where to link. NAME allows this tag to be a target of a hypertext link.

The difference between these subprograms is that http.anchor2 provides a target and therefore can be used for a frame.

Parameters

curl - the value for the HREF attribute.

ctext - the string that goes between the <A> and tags.

cname - the value for the NAME attribute.

cttarget - the value for the TARGET attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<A HREF="curl" NAME="cname" cattributes>ctext</A>  
<A HREF="curl" NAME="cname" TARGET = "ctarget" cattributes>ctext</A>
```

htp.appletopen, htp.appletclose

Syntax

```
htp.appletopen(
    ccode          in      varchar2
    cheight        in      number
    cwidth         in      number
    cattributes    in      varchar2  DEFAULT NULL);
htf.appletopen(ccode, cheight, cwidth, cattributes) return varchar2;

htp.appletclose;
htf.appletclose return varchar2;
```

Purpose

htp.appletopen generates the <APPLET> HTML tag, which begins the invocation of a Java applet. You close the applet invocation with htp.appletclose, which generates the </APPLET> HTML tag.

You can specify parameters to the Java applet using the [htp.param](#) procedure.

You have to use the cattributes parameter to specify the CODEBASE attribute because the PL/SQL cartridge does not know where to find the class files. The CODEBASE attribute specifies the virtual path containing the class files.

Parameters

ccode - the value for the CODE attribute, which specifies the name of the applet class.

cheight - the value for the HEIGHT attribute.

cwidth - the value for the WIDTH attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<APPLET CODE=ccode HEIGHT=cheight WIDTH=cwidth cattributes>
</APPLET>
```

`http.appletopen`, `http.appletclose`

Example

```
http.appletopen('testclass.class', 100, 200, 'CODEBASE="/ows-applets"')  
  
generates  
<APPLET CODE="testclass.class" height=100 width=200 CODEBASE="/ows-applets">
```

htp.area

Syntax

```
htp.area(  
    ccoords      in      varchar2  
    cshape       in      varchar2  DEFAULT NULL  
    chref        in      varchar2  DEFAULT NULL  
    cnohref      in      varchar2  DEFAULT NULL  
    ctarger      in      varchar2  DEFAULT NULL  
    cattributes  in      varchar2  DEFAULT NULL);  
htf.area(ccoords, cshape, chref, cnohref, ctarger, cattributes) return  
varchar2;
```

Purpose

Generates the <AREA> HTML tag, which defines a client-side image map. This tag defines areas within the image and destinations for the areas.

Parameters

ccoords - the value for the COORDS attribute.

cshape - the value for the SHAPE attribute.

chref - the value for the HREF attribute.

cnohref - if the value for this parameter is not NULL, the NOHREF attribute is added to the tag.

ctarger - the value for the TARGET attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<AREA COORDS="ccoords" SHAPE="cshape" HREF="chref" NOHREF TAR-  
GET="ctarget" cattributes>
```

http.base

Syntax

```
http.base(
    ctarget      in      varchar2  DEFAULT NULL
    cattributes  in      varchar2  DEFAULT NULL);
htf.base(ctarget, cattributes) return varchar2;
```

Purpose

Generates the <BASE> HTML tag, which records the URL of the document.

Parameters

ctarget - the value for the TARGET attribute, which establishes a window name to which all links in this document are targeted.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<BASE HREF="<current URL>" TARGET="ctarget" cattributes>
```

htp.basefont

Syntax

```
htp.basefont(nsize in integer);
htf.basefont(nsize) return varchar2;
```

Purpose

Generates the <BASEFONT> HTML tag, which specifies the base font size for a Web page.

Parameters

nsize - the value for the SIZE attribute.

Generates

```
<BASEFONT SIZE="nsize">
```

http.bgsound

Syntax

```
http.bgsound(
    csrc          in      varchar2
    cloop         in      varchar2  DEFAULT NULL
    cattributes   in      varchar2  DEFAULT NULL);
htf.bgsound(csrc, cloop, cattributes) return varchar2;
```

Purpose

Generates the <BGSOUND> HTML tag, which includes audio for a Web page.

Parameters

csrc - the value for the SRC attribute.

cloop - the value for the LOOP attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<BGSOUND SRC="csrc" LOOP="cloop" cattributes>
```

htp.big

Syntax

```
htp.big(
    ctext          in      varchar2
    cattributes   in      varchar2  DEFAULT NULL);
htf.big(ctext, cattributes) return varchar2;
```

Purpose

Generates the <BIG> and </BIG> tags, which direct the browser to render the text in a bigger font.

Parameters

ctext - the text that goes between the tags.

cattributes - other attributes to be included as-is in the tag.

Generates

<BIG *cattributes*>*ctext*</BIG>

http.blockquoteOpen, http.blockquoteClose

Syntax

```
http.blockquoteOpen (
    cnowrap      in      varchar2  DEFAULT NULL
    cclear       in      varchar2  DEFAULT NULL
    cattributes   in      varchar2  DEFAULT NULL);
htf.blockquoteOpen (cnowrap, cclear, cattributes) return varchar2;

http.blockquoteClose;
htf.blockquoteClose return varchar2;
```

Purpose

Generates the <BLOCKQUOTE> and </BLOCKQUOTE> tag, which mark a section of quoted text.

Parameters

- cnowrap - if the value for this parameter is not NULL, the NOWRAP attribute is added to the tag.
- cclear - the value for the CLEAR attribute.
- cattributes - other attributes to be included as-is in the tag.

Generates

```
<BLOCKQUOTE CLEAR="cclear" NOWRAP cattributes>
</BLOCKQUOTE>
```

htp.bodyOpen, htp.bodyClose

Syntax

```
htp.bodyOpen(  
    cbackground      in      varchar2  DEFAULT NULL  
    cattributes     in      varchar2  DEFAULT NULL);  
htf.bodyOpen(cbackground, cattributes) return varchar2;  
  
htp.bodyClose;  
htf.bodyClose return varchar2;
```

Purpose

Generates the <BODY> and </BODY> tags, which mark the body section of an HTML document.

Parameters

cbackground - the value for the BACKGROUND attribute, which specifies a graphic file to use for the background of the document.
cattributes - other attributes to be included as-is in the tag.

Generates

```
<BODY background="cbackground" cattributes>  
</BODY>
```

Example

```
htp.bodyOpen('/img/background.gif');  
generates:  
<BODY background="/img/background.gif">
```

http.bold

Syntax

```
http.bold (
    ctext      in      varchar2
    cattributes  in      varchar2  DEFAULT NULL);
htf.bold (ctext, cattributes) return varchar2;
```

Purpose

Generates the and tags, which direct the browser to display the text in boldface.

Parameters

ctext - the text that goes between the tags.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<B cattributes>ctext</B>
```

htp.center

Syntax

```
htp.center(ctext in varchar2);
htf.center(ctext in varchar2) return varchar2;
```

Purpose

Generates the <CENTER> and </CENTER> tags, which center a section of text within a Web page.

Parameters

ctext - the text to center.

Generates

<CENTER>*ctext*</CENTER>

htp.centerOpen, htp.centerClose

Syntax

```
htp.centerOpen;  
htf.centerOpen return varchar2;  
  
htp.centerClose;  
htf.centerClose return varchar2;
```

Purpose

Generates the <CENTER> and </CENTER> tags, which mark the section of text to center.

Parameters

None.

Generates

```
<CENTER>  
  
</CENTER>
```

htp.cite

Syntax

```
htp.cite (
    ctext          in      varchar2
    cattributes   in      varchar2  DEFAULT NULL);
htf.cite (ctext, cattributes) return varchar2;
```

Purpose

Generates the <CITE> and </CITE> tags, which direct the browser to render the text as citation.

Parameters

ctext - the text to render as citation.

cattributes - other attributes to be included as-is in the tag.

Generates

<CITE *cattributes*>*ctext*</CITE>

http.code

Syntax

```
http.code (
    ctext      in      varchar2
    cattributes      in      varchar2  DEFAULT NULL);
htf.code (ctext, cattributes) return varchar2;
```

Purpose

Generates the <CODE> and </CODE> tags, which direct the browser to render the text in monospace font.

Parameters

ctext - the text to render as code.

cattributes - other attributes to be included as-is in the tag.

Generates

<CODE *cattributes*>*ctext*</CODE>

htp.comment

Syntax

```
htp.comment (ctext in varchar2);
htf.comment (ctext in varchar2) return varchar2;
```

Purpose

Generates the comment tags.

Parameters

ctext - the comment.

Generates

```
<!-- ctext -->
```

http.dfn

Syntax

```
http.dfn(ctext in varchar2);
htf.dfn(ctext in varchar2) return varchar2;
```

Purpose

Generates the <DFN> and </DFN> tags, which direct the browser to render the text in italics.

Parameters

ctext - the text to render in italics.

Generates

<DFN>*ctext*</DFN>

http.dirlistOpen, http.dirlistClose

Syntax

```
http.dirlistOpen;  
htf.dirlistOpen return varchar2;  
  
http.dirlistClose;  
htf.dirlistClose return varchar2;
```

Purpose

Generates the <DIR> and </DIR> tags, which create a directory list section. A directory list presents a list of items that contains up to 20 characters. Items in this list are typically arranged in columns, typically 24 characters wide. The tag or [http.listItem](#) must appear directly after you use this tag to define the items in the list.

Parameters

None.

Generates

```
<DIR>  
  
</DIR>
```

http.div

Syntax

```
http.div(  
    calign      in      varchar2  DEFAULT NULL  
    cattributes in      varchar2  DEFAULT NULL);  
htf.div(calign, cattributes) return varchar2;
```

Purpose

Generates the <DIV> tag, which creates document divisions.

Parameters

calign - the value for the ALIGN attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

<DIV ALIGN="*calign*" *cattributes*>

htp.dlistOpen, htp.dlistClose

Syntax

```
htp.dlistOpen (
    cclear      in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.dlistOpen (cclear, cattributes) return varchar2;

htp.dlistClose;
htf.dlistClose return varchar2;
```

Purpose

Generates the <DL> and </DL> tags, which create a definition list. A definition list looks like a glossary: it contains terms and definitions. Terms are inserted using [htp.dlistTerm](#), and definitions are inserted using [htp.dlistDef](#).

Parameters

- cclear - the value for the CLEAR attribute.
- cattributes - other attributes to be included as-is in the tag.

Generates

```
<DL CLEAR="cclear" cattributes>
</DL>
```

http.dlistDef

Syntax

```
http.dlistDef(  
    ctext      in      varchar2  DEFAULT NULL  
    cclear     in      varchar2  DEFAULT NULL  
    cattributes in      varchar2  DEFAULT NULL);  
htf.dlistDef(ctext, cclear, cattributes) return varchar2;
```

Purpose

Generates the <DD> tag, which is used to insert definitions of terms. This tag is used in the context of the definition list <DL>, where terms are tagged with <DT> and definitions are tagged with <DD>.

Parameters

- ctext - the definition for the term.
- cclear - the value for the CLEAR attribute.
- cattributes - other attributes to be included as-is in the tag.

Generates

```
<DD CLEAR="cclear" cattributes>ctext
```

htp.dlistTerm

Syntax

```
htp.dlistTerm (
    ctext          in      varchar2  DEFAULT NULL
    cclear         in      varchar2  DEFAULT NULL
    cattributes   in      varchar2  DEFAULT NULL);
htf.dlistTerm (ctext, cclear, cattributes) return varchar2;
```

Purpose

Generates the <DT> tag, which defines a term in a definition list <DL>.

Parameters

ctext - the term.

cclear - the value for the CLEAR attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

<DT CLEAR="*cclear*" *cattributes*>*ctext*

http.download_file

Syntax

```
http.download_file(sFileName in varchar2)
http.download_file(bCompress in boolean DEFAULT FALSE);
```

Purpose

After you have uploaded files to the database, you can download them, delete them from the database, and read and write their attributes.

Parameters

sFileName - file to be downloaded.

bCompress - file to be compressed.

Returns

The downloaded file.

http.get_download_files_list

Syntax

```
http.get_download_files_list(sFileName out varchar2)
http.get_download_files_list(bCompress out binary_integer);
```

Purpose

After you have downloaded files to the database, you need to get the files.

Parameters

sFileName - file to get.
bCompress - file to compress.

Returns

The downloaded file.

http.emphasis, http.em

Syntax

```
http.em (
    ctext      in      varchar2
    cattributes      in      varchar2  DEFAULT NULL);
htf.em (ctext, cattributes) return varchar2;

http.emphasis (
    ctext      in      varchar2
    cattributes      in      varchar2  DEFAULT NULL);
htf.emphasis (ctext, cattributes) return varchar2;
```

Purpose

Generates the and tags, which define text to be emphasized.

Parameters

ctext - the text to emphasize.

cattributes - other attributes to be included as-is in the tag.

Generates

<EM cattributes>ctext

htf.escape_sc

Syntax

```
htf.escape_sc(ctext in varchar2) return varchar2;
htp.escape_sc(ctext in varchar2);
```

Purpose

Replaces characters that have special meaning in HTML with their escape sequences. The following characters are converted:

From	To
&	&
"	"
<	<
>	>

Note that the procedure version of this subprogram does the same thing as htp.prints and htp.ps.

Parameters

ctext - the string to convert.

Returns

The converted string.

htf.escape_url

Syntax

```
htf.escape_url(p_url in varchar2) return varchar2;
```

Purpose

Replaces characters that have special meaning in HTML and HTTP with their escape sequences. The following characters are converted:

From	To
&	&
"	"
<	<
>	>
%	%25

Parameters

p_url - the string to convert.

Returns

The converted string.

htp.fontOpen, htp.fontClose

Syntax

```
htp.fontOpen(  
    ccolor      in      varchar2  DEFAULT NULL  
    cface       in      varchar2  DEFAULT NULL  
    csize       in      varchar2  DEFAULT NULL  
    cattributes in      varchar2  DEFAULT NULL);  
htf.fontOpen(ccolor, cface, csize, cattributes) return varchar2;  
  
htp.fontClose;  
htf.fontClose return varchar2;
```

Purpose

Generates the and tags, which mark a section of text with the specified font characteristics.

Parameters

- ccolor - the value for the COLOR attribute.
- cface - the value for the FACE attribute.
- csize - the value for the SIZE attribute.
- cattributes - other attributes to be included as-is in the tag.

Generates

```
<FONT COLOR="ccolor" FACE="cface" SIZE="csize" cattributes">  
</FONT>
```

htf.format_cell

Syntax

```
htf.format_cell (
    columnValue      in      varchar2
    format_numbers   in      varchar2  DEFAULT NULL) return
varchar2;
```

Purpose

Formats column values inside an HTML table using htf.tableData. Allows you to have a finer control over HTML tables.

Parameters

columnValue - the value that needs to be formatted in an HTML table.

format_numbers - the format in which numeric data should be displayed. If the value of this parameter is not Null, number fields are right-justified and rounded to two decimal places.

Generates

<TD>*columnValue*</TD>

htp.formCheckbox

Syntax

```
htp.formCheckbox (
    cname      in      varchar2
    cvalue     in      varchar2 DEFAULT 'on'
    cchecked   in      varchar2 DEFAULT NULL
    cattributes in      varchar2 DEFAULT NULL);
htf.formCheckbox (cname, cvalue, cchecked, cattributes) return varchar2;
```

Purpose

Generates the <INPUT> tag with TYPE="checkbox", which inserts a checkbox element in a form. A checkbox element is a button that the user can toggle on or off.

Parameters

cname - the value for the NAME attribute.

cvalue - the value for the VALUE attribute.

cchecked - if the value for this parameter is not NULL, the CHECKED attribute is added to the tag.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<INPUT TYPE="checkbox" NAME="cname" VALUE="cvalue" CHECKED cattributes>
```

http.formOpen, http.formClose

Syntax

```
http.formOpen (
    curl          in      varchar2
    cmethod      in      varchar2  DEFAULT 'POST'
    ctarget       in      varchar2
    cenctype     in      varchar2  DEFAULT NULL
    cattributes   in      varchar2  DEFAULT NULL);
htf.formOpen (curl, cmethod, ctarget, cenctype, cattributes) return
varchar2;

http.formClose;
htf.formClose return varchar2;
```

Purpose

Generates the <FORM> and </FORM> tags, which create a form section in an HTML document.

Parameters

curl - the URL of the WRB cartridge or CGI script to which the contents of the form is sent. This parameter is required.

cmethod - the value for the METHOD attribute. The value can be “GET” or “POST”.

ctarget - the value for the TARGET attribute.

cenctype - the value for the ENCTYPE attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<FORM ACTION="curl" METHOD="cmethod" TARGET="ctarget" ENCTYPE="cen-
ctype" cattributes>
```

```
</FORM>
```

htp.formHidden

Syntax

```
htp.formHidden (
    cname          in      varchar2
    cvalue         in      varchar2  DEFAULT NULL
    cattributes   in      varchar2  DEFAULT NULL);
htf.formHidden (cname, cvalue, cattributes) return varchar2;
```

Purpose

Generates the <INPUT> tag with TYPE="hidden", which inserts a hidden form element. This element is not seen by the user and is used to submit additional values to the script.

Parameters

cname - the value for the NAME attribute.

cvalue - the value for the VALUE attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<INPUT TYPE="hidden" NAME="cname" VALUE="cvalue" cattributes>
```

http.formImage

Syntax

```
http.formImage (
    cname      in      varchar2
    csrc       in      varchar2
    calign     in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.formImage (cname, csrc, calign, cattributes) return varchar2;
```

Purpose

Generates the <INPUT> tag with TYPE="image", which creates an image field on which the user can click and cause the form to be submitted immediately. The coordinates of the selected point are measured in pixels, and returned (along with other contents of the form) in two name-/value pairs. The x coordinate is submitted under the name of the field with ".x" appended, and the y coordinate with the ".y" appended. Any VALUE attribute is ignored.

Parameters

cname - the VALUE for the NAME attribute.
csrc - the value for the SRC attribute, which specifies the image file.
calign - the value for the ALIGN attribute.
cattributes - other attributes to be included as-is in the tag.

Generates

```
<INPUT TYPE="image" NAME="cname" SRC="csrc" ALIGN="calign" cattributes>
```

htp.formPassword

Syntax

```
htp.formPassword (
    cname      in      varchar2
    csize      in      varchar2
    cmaxlength in      varchar2 DEFAULT NULL
    cvalue     in      varchar2 DEFAULT NULL
    cattributes in      varchar2 DEFAULT NULL);
htf.formPassword (cname, csize, cmaxlength, cvalue, cattributes) return
varchar2;
```

Purpose

Generates the <INPUT> tag with TYPE="password", which creates a single-line text entry field. When the user enters text in the field, each character is represented by one asterisk. This is usually used for entering passwords.

Parameters

cname - the value for the NAME attribute.
csize - the value for the SIZE attribute.
cmaxlength - the value for the MAXLENGTH attribute.
cvalue - the value for the VALUE attribute.
cattributes - other attributes to be included as-is in the tag.

Generates

```
<INPUT TYPE="password" NAME="cname" SIZE="csize" MAXLENGTH="cmaxlength" VALUE="cvalue" cattributes>
```

http.formRadio

Syntax

```
http.formRadio (
    cname      in      varchar2
    cvalue     in      varchar2
    cchecked   in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.formRadio (cname, cvalue, cchecked, cattributes) return varchar2;
```

Purpose

Generates the <INPUT> tag with TYPE= "radio", which creates a radio button on the HTML form.

Within a set of radio buttons, the user can select only one button. Each radio button in the same set should have the same name, but different value. The selected radio button generates a name/value pair.

Parameters

cname - the value for the NAME attribute.

cvalue - the value for the VALUE attribute.

cchecked - if the value for this parameter is not NULL, the CHECKED attribute is added to the tag.

cattributes - other attributes to be included as-is in the tag.

Generates

<INPUT TYPE= "radio" NAME= "cname" VALUE= "cvalue" CHECKED cattributes>

htp.formReset

Syntax

```
htp.formReset (
    cvalue      in      varchar2  DEFAULT 'Reset'
    cattributes in      varchar2  DEFAULT NULL);
htf.formReset (cvalue, cattributes) return varchar2;
```

Purpose

Generates the <INPUT> tag with TYPE="reset", which creates a button that, when clicked, resets all the form fields to their initial values.

Parameters

cvalue - the value for the VALUE attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<INPUT TYPE="reset" VALUE="cvalue" cattributes>
```

http.formSelectOpen, http.formSelectClose

Syntax

```
http.formSelectOpen (
    cname      in      varchar2
    cprompt    in      varchar2  DEFAULT NULL
    nsize      in      integer  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.formSelectOpen (cname, cprompt, nsize, cattributes) return varchar2;

http.formSelectClose;
htf.formSelectClose return varchar2;
```

Purpose

Generates the <SELECT> and </SELECT> tags, which create a Select form element. A Select form element is a listbox, from which the user can select one or more values. The values are inserted using [http.formSelectOption](#).

Parameters

cname - the value for the NAME attribute.
cprompt - the string preceding the list box.
nsize - the value for the SIZE attribute.
cattributes - other attributes to be included as-is in the tag.

Generates

```
cprompt <SELECT NAME="cname" SIZE="nsize" cattributes>
</SELECT>
```

Example

```
http.formSelectOpen('greatest_player';
    'Pick the greatest player:');
http.formSelectOption('Messier');
http.formSelectOption('Howe');
http.formSelectOption('Gretzky');
http.formSelectClose;
```

Generates:

```
Pick the greatest player:  
<SELECT NAME="greatest_player">  
<OPTION>Messier  
<OPTION>Howe  
<OPTION>Gretzky  
</SELECT>
```

http.formSelectOption

Syntax

```
http.formSelectOption (
    cvalue      in      varchar2
    cselected   in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.formSelectOption (cvalue, cselected, cattributes) return varchar2;
```

Purpose

Generates the <OPTION> tag, which represents one choice in a Select element.

Parameters

cvalue - the text for the option.

cselected - if the value for this parameter is not NULL, the SELECTED attribute is added to the tag.

cattributes - other attributes to be included as-is in the tag.

Generates

<OPTION SELECTED *cattributes*>*cvalue*

Example

See [http.formSelectOpen](#), [http.formSelectClose](#)

htp.formSubmit

Syntax

```
htp.formSubmit (
    cname      in      varchar2  DEFAULT NULL
    cvalue     in      varchar2  DEFAULT 'Submit'
    cattributes in      varchar2  DEFAULT NULL);
htf.formSubmit (cname, cvalue, cattributes) return varchar2;
```

Purpose

Generates the <INPUT> tag with TYPE="submit", which creates a button that, when clicked, submits the form.

If the button has a NAME attribute, the button contributes a name/value pair to the submitted data.

Parameters

cname - the value for the NAME attribute.

cvalue - the value for the VALUE attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<INPUT TYPE="submit" NAME="cname" VALUE="cvalue" cattributes>
```

http.formText

Syntax

```
http.formText (
    cname      in      varchar2
    csize       in      varchar2  DEFAULT NULL
    cmaxlength in      varchar2  DEFAULT NULL
    cvalue      in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.formText (cname, csize, cmaxlength, cvalue, cattributes) return
varchar2;
```

Purpose

Generates the <INPUT> tag with TYPE="text", which creates a field for a single line of text.

Parameters

- cname - the value for the NAME attribute.
- csize - the value for the SIZE attribute.
- cmaxlength - the value for the MAXLENGTH attribute.
- cvalue - the value for the VALUE attribute.
- cattributes - other attributes to be included as-is in the tag.

Generates

```
<INPUT TYPE="text" NAME="cname" SIZE="csize" MAXLENGTH="cmaxlength"
VALUE="cvalue" cattributes>
```

http.formTextarea, http.formTextarea2

Syntax

```
http.formTextarea (
    cname      in      varchar2
    nrows      in      integer
    ncolumns   in      integer
    calign     in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.formTextarea (cname, nrows, ncolumns, calign, cattributes) return
varchar2;

http.formTextarea2 (
    cname      in      varchar2
    nrows      in      integer
    ncolumns   in      integer
    calign     in      varchar2  DEFAULT NULL
    cwrap      in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.formTextarea2 (cname, nrows, ncolumns, calign, cwrap, cattributes)
return varchar2;
```

Purpose

Generates the <TEXTAREA> tag, which creates a text field that has no predefined text in the text area. This field is used to enable the user to enter several lines of text.

The difference between these subprograms is that http.formTextarea2 has the *cwrap* parameter, which specifies a wrap style.

Parameters

cname - the value for the NAME attribute.

nrows - the value for the ROWS attribute. This is an integer.

ncolumns - the value for the COLS attribute. This is an integer.

calign - the value for the ALIGN attribute.

cwrap - the value for the WRAP attribute.

cattributes - other attributes to be included as-is in the tag.

`htp.formTextarea`, `htp.formTextarea2`

Generates

```
<TEXTAREA NAME="cname" ROWS="nrows" COLS="ncolumns" ALIGN="calign"  
cattributes></TEXTAREA>
```

```
<TEXTAREA NAME="cname" ROWS="nrows" COLS="ncolumns" ALIGN="calign"  
WRAP="cwrap" cattributes></TEXTAREA>
```

htp.formTextareaOpen, htp.formTextareaOpen2, htp.formTextareaClose

Syntax

```
htp.formTextareaOpen (
    cname      in      varchar2
    nrows      in      integer
    ncolumns   in      integer
    calign     in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.formTextareaOpen (cname, nrows, ncolumns, calign, cattributes)
return varchar2;

htp.formTextareaOpen2(
    cname      in      varchar2
    nrows      in      integer
    ncolumns   in      integer
    calign     in      varchar2  DEFAULT NULL
    cwrap      in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.formTextareaOpen2(cname, nrows, ncolumns, calign, cwrap,
cattributes) return varchar2;

htp.formTextareaClose;
htf.formTextareaClose return varchar2;
```

Purpose

Generates the <TEXTAREA> and </TEXTAREA> tags, which creates a text area form element.

The difference between the two open subprograms is that htp.formTextareaOpen2 has the *cwrap* parameter, which specifies a wrap style.

Parameters

cname - the value for the NAME attribute.

nrows - the value for the ROWS attribute. This is an integer.

ncolumns - the value for the COLS attribute. This is an integer.

calign - the value for the ALIGN attribute.

`cwrap` - the value for the WRAP attribute.

`cattributes` - other attributes to be included as-is in the tag.

Generates

```
<TEXTAREA NAME="cname" ROWS="nrows" COLS="ncolumns" ALIGN="calign"  
cattributes>  
  
<TEXTAREA NAME="cname" ROWS="nrows" COLS="ncolumns" ALIGN="calign"  
WRAP = "cwrap" cattributes>  
  
</TEXTAREA>
```

htp.frame

Syntax

```
htp.frame(
    csrc          in      varchar2
    cname         in      varchar2 DEFAULT NULL
    cmarginwidth  in      varchar2 DEFAULT NULL
    cmarginheight in      varchar2 DEFAULT NULL
    cscrolling    in      varchar2 DEFAULT NULL
    cnoresize     in      varchar2 DEFAULT NULL
    cattributes   in      varchar2 DEFAULT NULL);
htf.frame(csrc, cname, cmarginwidth, cmarginheight, cscrolling,
cnoresize, cattributes) return varchar2;
```

Purpose

Generates the <FRAME> tag, which defines the characteristics of a frame created by a <FRAMESET> tag.

Parameters

- csrc - the URL to display in the frame.
- cname - the value for the NAME attribute.
- cmarginwidth - the value for the MARGINWIDTH attribute.
- cmarginheight - the value for the MARGINHEIGHT attribute.
- cscrolling - the value for the SCROLLING attribute.
- noresize - if the value for this parameter is not NULL, the NORESIZE attribute is added to the tag.
- cattributes - other attributes to be included as-is in the tag.

Generates

```
<FRAME SRC="csrc" NAME="cname" MARGINWIDTH="cmarginwidth" MARGINHEIGHT="cmarginheight" SCROLLING="cscrolling" NORESIZE cattributes>
```

http.framesetOpen, http.framesetClose

Syntax

```
http.framesetOpen(  
    crows      in      varchar2  DEFAULT NULL  
    ccols      in      varchar2  DEFAULT NULL  
    cattributes in      varchar2  DEFAULT NULL);  
htf.framesetOpen(crows, ccols, cattributes) return varchar2;  
  
http.framesetClose;  
htf.framesetClose return varchar2;
```

Purpose

Generates the <FRAMESET> and </FRAMESET> tags, which define a frameset section.

Parameters

- crows - the value for the ROWS attribute.
- ccols - the value for the COLS attribute.
- cattributes - other attributes to be included as-is in the tag.

Generates

```
<FRAMESET ROWS="crows" COLS="ccols" cattributes>  
</FRAMESET>
```

htp.headOpen, htp.headClose

Syntax

```
htp.headOpen;  
htf.headOpen return varchar2;  
  
htp.headClose;  
htf.headClose return varchar2;
```

Purpose

Generates the <HEAD> and </HEAD> tags, which mark the HTML document head section.

Generates

```
<HEAD>  
  
</HEAD>
```

http.header

Syntax

```
http.header (
    nsize      in      integer
    cheader   in      varchar2
    calign    in      varchar2  DEFAULT NULL
    cnowrap   in      varchar2  DEFAULT NULL
    cclear    in      varchar2  DEFAULT NULL
    cattributes  in      varchar2  DEFAULT NULL);
htf.header (nsize, cheader, calign, cnowrap, cclear, cattributes)
return varchar2;
```

Purpose

Generates opening heading tags (<H1> to <H6>) and their corresponding closing tags (</H1> to </H6>).

Parameters

nsize - the heading level. This is an integer between 1 and 6.
cheader - the text to display in the heading.
calign - the value for the ALIGN attribute.
cnowrap - the value for the NOWRAP attribute.
cclear - the value for the CLEAR attribute.
cattributes - other attributes to be included as-is in the tag.

Generates

```
<Hnsize ALIGN="calign" NOWRAP CLEAR="cclear" cattributes>cheader</Hnsize>
```

Example

```
http.header (1,'Overview');
```

produces:

```
<H1>Overview</H1>
```

htp.htmlOpen, htp.htmlClose

Syntax

```
htp.htmlOpen;  
htf.htmlOpen return varchar2;  
  
htp.htmlClose;  
htf.htmlClose return varchar2;
```

Purpose

Generates the <HTML> and </HTML> tags, which mark the beginning and the end of an HTML document.

Parameters

None.

Generates

```
<HTML>  
  
</HTML>
```

http.img, http.img2

Syntax

```
http.img (
    curl          in      varchar2  DEFAULT NULL
    calign        in      varchar2  DEFAULT NULL
    calt          in      varchar2  DEFAULT NULL
    cismap        in      varchar2  DEFAULT NULL
    cattributes   in      varchar2  DEFAULT NULL);
htf.img (curl, calign, calt, cismap, cattributes) return varchar2;

http.img2(
    curl          in      varchar2  DEFAULT NULL
    calign        in      varchar2  DEFAULT NULL
    calt          in      varchar2  DEFAULT NULL
    cismap        in      varchar2  DEFAULT NULL
    cusemap       in      varchar2  DEFAULT NULL
    cattributes   in      varchar2  DEFAULT NULL);
htf.img2(curl, calign, calt, cismap, cusemap, cattributes) return
varchar2;
```

Purpose

Generates the tag, which directs the browser to load an image onto the HTML page.

The difference between these subprograms is that http.img2 takes the *cusemap* parameter.

Parameters

curl - the value for the SRC attribute.

calign - the value for the ALIGN attribute.

calt - the value for the ALT attribute, which specifies alternative text to display if the browser does not support images.

cismap - if the value for this parameter is not NULL, the ISMAP attribute is added to the tag. The attribute indicates that the image is an imagemap.

cusemap - the value for the USEMAP attribute, which specifies a client-side image map.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<IMG SRC="curl" ALIGN="calign" ALT="calt" ISMAP cattributes>
```

```
<IMG SRC="curl" ALIGN="calign" ALT="calt" ISMAP USEMAP="cusemap" ctributes>
```

http.isindex

Syntax

```
http.isindex (
    cprompt      in      varchar2      DEFAULT NULL
    curl         in      varchar2      DEFAULT NULL);
htf.isindex (cprompt, curl) return varchar2;
```

Purpose

Creates a single entry field with a prompting text, such as “*enter value*,” then sends that value to the URL of the page or program.

Parameters

- cprompt - the value for the PROMPT attribute.
- curl - the value for the HREF attribute.

Generates

```
<ISINDEX PROMPT="cprompt" HREF="curl">
```

htp.italic

Syntax

```
htp.italic (
    ctext          in      varchar2
    cattributes   in      varchar2  DEFAULT NULL);
htf.italic (ctext, cattributes) return varchar2;
```

Purpose

Generates the `<I>` and `</I>` tags, which direct the browser to render the text in italics.

Parameters

`ctext` - the text to be rendered in italics.

`cattributes` - other attributes to be included as-is in the tag.

Generates

`<I cattributes>ctext</I>`

http.keyboard, http.kbd

Syntax

```
http.keyboard (
    ctext      in      varchar2
    cattributes      in      varchar2  DEFAULT NULL);
htf.keyboard (ctext, cattributes) return varchar2;

http.kbd (
    ctext      in      varchar2
    cattributes      in      varchar2  DEFAULT NULL);
htf.kbd (ctext, cattributes) return varchar2;
```

Purpose

Generates the <KBD> and </KBD> tags, which direct the browser to render the text in monospace. These subprograms do the same thing.

Parameters

ctext - the text to render in monospace.

cattributes - other attributes to be included as-is in the tag.

Generates

<KBD *cattributes*>*ctext*</KBD>

http.line, http.hr

Syntax

```
http.line (
    cclear      in      varchar2  DEFAULT NULL
    csrcc      in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.line (cclear, csrcc, cattributes) return varchar2;

http.hr (
    cclear      in      varchar2  DEFAULT NULL
    csrcc      in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.hr (cclear, csrcc, cattributes) return varchar2;
```

Purpose

Generates the <HR> tag, which generates a line in the HTML document.

Parameters

cclear - the value for the CLEAR attribute.

csrcc - the value for the SRC attribute, which specifies a custom image as the source of the line.

cattributes - other attributes to be included as-is in the tag.

Generates

<HR CLEAR="*cclear*" SRC="*csrcc*" *cattributes*>

http.linkRel

Syntax

```
http.linkRel (
    crel      in      varchar2
    curl      in      varchar2
    ctitle    in      varchar2  DEFAULT NULL);
htf.linkRel (crel, curl, ctitle) return varchar2;
```

Purpose

Generates the <LINK> tag with the REL attribute, which gives the relationship described by the hypertext link from the anchor to the target. This is only used when the HREF attribute is present. This tag indicates a relationship between documents, but does not create a link. To create a link, use [http.anchor](#), [http.anchor2](#).

Parameters

- crel - the value for the REL attribute.
- curl - the value for the HREF attribute.
- ctitle - the value for the TITLE attribute.

Generates

```
<LINK REL="crel" HREF="curl" TITLE="ctitle">
```

htp.linkRev

Syntax

```
htp.linkRev (
    crev          in      varchar2
    curl          in      varchar2
    ctitle        in      varchar2  DEFAULT NULL);
htf.linkRev (crev, curl, ctitle) return varchar2;
```

Purpose

Generates the <LINK> tag with the REV attribute, which gives the relationship described by the hypertext link from the target to the anchor. This is the opposite of [htp.linkRel](#). This tag indicates a relationship between documents, but does not create a link. To create a link, use [htp.anchor](#), [htp.anchor2](#).

Parameters

- crev - the value for the REV attribute.
- curl - the value for the HREF attribute.
- ctitle - the value for the TITLE attribute.

Generates

```
<LINK REV="crev" HREF="curl" TITLE="ctitle">
```

http.listHeader

Syntax

```
http.listHeader (
    ctext      in      varchar2
    cattributes      in      varchar2  DEFAULT NULL);
htf.listHeader (ctext, cattributes) return varchar2;
```

Purpose

Generates the <LH> and </LH> tags, which print an HTML tag at the beginning of the list.

Parameters

ctext - the text to place between <LH> and </LH>.

cattributes - other attributes to be included as-is in the tag.

Generates

<LH *cattributes*>*ctext*</LH>

htp.listingOpen, htp.listingClose

Syntax

```
htp.listingOpen;  
htf.listingOpen return varchar2;  
  
htp.listingClose;  
htf.listingClose return varchar2;
```

Purpose

Generates the <LISTING> and </LISTING> tags, which mark a section of fixed-width text in the body of an HTML page.

Parameters

None.

Generates

```
<LISTING>  
  
</LISTING>
```

http.listItem

Syntax

```
http.listItem (
    ctext          in      varchar2  DEFAULT NULL
    cclear         in      varchar2  DEFAULT NULL
    cdingbat      in      varchar2  DEFAULT NULL
    csrc          in      varchar2  DEFAULT NULL
    cattributes   in      varchar2  DEFAULT NULL);
htf.listItem (ctext, cclear, cdingbat, csrc, cattributes) return
varchar2;
```

Purpose

Generates the tag, which indicates a list item.

Parameters

ctext - the text for the list item.

cclear - the value for the CLEAR attribute.

cdingbat - the value for the DINGBAT attribute.

csrc - the value for the SRC attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

<LI CLEAR="*cclear*" DINGBAT="*cdingbat*" SRC="*csrc*" *cattributes*>*ctext*

http.mailto

Syntax

```
http.mailto (
    caddress      in      varchar2
    ctext         in      varchar2
    cname         in      varchar2
    cattributes   in      varchar2  DEFAULT NULL);
htf.mailto (caddress, ctext, cname, cattributes) return varchar2;
```

Purpose

Generates the <A> tag with the HREF set to 'mailto' prepended to the mail address argument.

Parameters

caddress - the email address of the recipient.
ctext - the clickable portion of the link.
cname - the value for the NAME attribute.
cattributes - other attributes to be included as-is in the tag.

Generates

```
<A HREF="mailto:caddress" NAME="cname" cattributes>ctext</A>
```

Example

```
http.mailto('pres@white_house.gov', 'Send Email to the President');

generates:

<A HREF="mailto:pres@white_house.gov">Send Email to the President</A>
```

http.mapOpen, http.mapClose

Syntax

```
http.mapOpen(
    cname      in      varchar2
    cattributes in      varchar2  DEFAULT NULL);
htf.mapOpen(cname, cattributes) return varchar2;

http.mapClose;
htf.mapClose return varchar2;
```

Purpose

Generates the <MAP> and </MAP> tags, which mark a set of regions in a client-side image map.

Parameters

cname - the value for the NAME attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<MAP NAME="cname" cattributes>
</MAP>
```

htp.menulistOpen, htp.menulistClose

Syntax

```
htp.menulistOpen;  
htf.menulistOpen return varchar2;  
  
htp.menulistClose;  
htf.menulistClose return varchar2;
```

Purpose

Generates the <MENU> and </MENU> tags, which create a list that presents one line per item. The items in the list appear more compact than an unordered list. The [htp.listItem](#) defines the list items in a menu list.

Parameters

None.

Generates

```
<MENU>  
  
</MENU>
```

http.meta

Syntax

```
http.meta (
    chttp_equiv      in      varchar2
    cname           in      varchar2
    ccontent        in      varchar2);
htf.meta (chttp_equiv, cname, ccontent) return varchar2;
```

Purpose

Generates the <META> tag, which enables you to embed meta-information about the document and also specify values for HTTP headers. For example, you can specify the expiration date, keywords, and author name.

Parameters

chttp_equiv - the value for the HTTP-EQUIV attribute.
cname - the value for the NAME attribute.
ccontent - the value for the CONTENT attribute.

Generates

```
<META HTTP-EQUIV="chttp_equiv" NAME="cname" CONTENT="ccontent">
```

Example

```
http.meta ('Refresh', NULL, 120);
```

generates:

```
<META HTTP-EQUIV="Refresh" CONTENT=120>
```

On some Web browsers, this causes the current URL to be reloaded automatically every 120 seconds.

http.nl, http.br

Syntax

```
http.nl (
    cclear      in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.nl (cclear, cattributes) return varchar2;

http.br (
    cclear      in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.br (cclear, cattributes) return varchar2;
```

Purpose

Generates the
 tag, which begins a new line of text.

Parameters

cclear - the value for the CLEAR attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

<BR CLEAR="*cclear*" *cattributes*>

http.nobr

Syntax

```
http.nobr(ctext in varchar2);
htf.nobr(ctext) return varchar2;
```

Purpose

Generates the <NOBR> and </NOBR> tags, which turn off line-breaking in a section of text.

Parameters

ctext - the text that is to be rendered on one line.

Generates

<NOBR>*ctext*</NOBR>

htp.noframesOpen, htp.noframesClose

Syntax

```
htp.noframesOpen  
htf.noframesOpen return varchar2;  
  
htp.noframesClose  
htf.noframesClose return varchar2;
```

Purpose

Generates the <NOFRAMES> and </NOFRAMES> tags, which mark a no-frames section.

Parameters

None.

Generates

```
<NOFRAMES>  
  
</NOFRAMES>
```

See Also

[htp.frame](#), [htp.framesetOpen](#), [htp.framesetClose](#)

http.olistOpen, http.olistClose

Syntax

```
http.olistOpen (
    cclear      in      varchar2  DEFAULT NULL
    cwrap       in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.olistOpen (cclear, cwrap, cattributes) return varchar2;

http.olistClose;
htf.olistClose return varchar2;
```

Purpose

Generates the and tags, which define an ordered list. An ordered list presents a list of numbered items. The numbered items are added using [http.list-Item](#).

Parameters

cclear - the value for the CLEAR attribute.
cwrap - the value for the WRAP attribute.
cattributes - other attributes to be included as-is in the tag

Generates

```
<OL CLEAR="cclear" WRAP="cwrap" cattributes>
</OL>
```

htp.para, htp.paragraph

Syntax

```
htp.para;  
htf.para return varcahr2;  
  
htp.paragraph (  
    calign      in      varchar2  DEFAULT NULL  
    cnowrap     in      varchar2  DEFAULT NULL  
    cclear      in      varchar2  DEFAULT NULL  
    cattributes in      varchar2  DEFAULT NULL);  
htf.paragraph (calign, cnowrap, cclear, cattributes) return varchar2;
```

Purpose

Generates the <P> tag, which indicates that the text that comes after the tag is to be formatted as a paragraph.

htp.paragraph enables you add attributes to the tag.

Parameters

calign - the value for the ALIGN attribute.

cnowrap - if the value for this parameter is not NULL, the NOWRAP attribute is added to the tag.

cclear - the value for the CLEAR attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

<P>

<P ALIGN="*calign*" NOWRAP CLEAR="*cclear*" *cattributes*>

htf.param

Syntax

```
htf.param(  
    cname      in      varchar2  
    cvalue     in      varchar2);  
htf.param(cname, cvalue) return varchar2;
```

Purpose

Generates the <PARAM> tag, which specifies parameter values for Java applets. The values can reference HTML variables.

To invoke a Java applet from a Web page, use `htf.appletopen` to begin the invocation. Use one `htf.param` for each desired name-value pair, and use `htf.appletclose` to end the applet invocation.

Parameters

`cname` - the value for the NAME attribute.

`cvalue` - the value for the VALUE attribute.

Generates

<PARAM NAME=*cname* VALUE="*cvalue*">

htp.plaintext

Syntax

```
htp.plaintext(
    ctext          in      varchar2
    cattributes   in      varchar2  DEFAULT NULL);
htf.plaintext(ctext, cattributes) return varchar2;
```

Purpose

Generates the <PLAINTEXT> and </PLAINTEXT> tags, which direct the browser to render the text they surround in fixed-width type.

Parameters

ctext - the text to be rendered in fixed-width font.

cattributes - other attributes to be included as-is in the tag.

Generates

<PLAINTEXT *cattributes*>*ctext*</PLAINTEXT>

http.preOpen, http.preClose

Syntax

```
http.preOpen (
    cclear      in      varchar2  DEFAULT NULL
    cwidth      in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.preOpen (cclear, cwidth, cattributes) return varchar2;

http.preClose;
htf.preClose return varchar2;
```

Purpose

Generates the <PRE> and </PRE> tags, which mark a section of preformatted text in the body of the HTML page.

Parameters

- cclear - the value for the CLEAR attribute.
- cwidth - the value for the WIDTH attribute.
- cattributes - other attributes to be included as-is in the tag.

Generates

```
<PRE CLEAR="cclear" WIDTH="cwidth" cattributes>
</PRE>
```

htp.print, htp.prn

Syntax

```
htp.print (cbuf in varchar2);
htp.print (dbuf in date);
htp.print (nbuf in number);

htp.prn (cbuf in varchar2);
htp.prn (dbuf in date);
htp.prn (nbuf in number);
```

Purpose

`htp.print` generates the specified parameter as a string terminated with the `\n` newline character.

Note that the `\n` character is not the same as `
`. The `\n` character is used to format the HTML source; it does not affect how the browser renders the HTML source. Use `
` to control how the browser renders the HTML source.

`htp.prn` generates the specified parameter as a string. Unlike `htp.print`, the string is not terminated with the `\n` newline character.

These subprograms are procedures only, they do not come as functions.

Parameters

`cbuf`, `dbuf`, `nbuf` - the string to generate.

Generates

`htp.print` - a string terminated with a newline.

`htp.prn` - the specified string, not terminated with a newline.

http.prints, http.ps

Syntax

```
http.prints(ctext in varchar2);
http.ps(ctext in varchar2);
```

Purpose

Both these subprograms generate a string and replace all occurrences of the following characters with the corresponding escape sequence.

Replaces this character	with this escape sequence
<	<
>	>
"	"
&	&

If not replaced, the special characters would be interpreted as HTML control characters and would produce garbled output. This procedure is the same as `http.prn` but with the character substitution.

These subprograms are procedures only, they are not available as functions. If you need a string conversion function, use [htf.escape_sc](#).

Parameters

ctext - the string in which to perform character substitution.

Generates

A string.

htp.s

Syntax

```
htp.s(
    ctext      in      varchar2
    cattributes  in      varchar2  DEFAULT NULL);
htf.s(ctext, cattributes) return varchar2;
```

Purpose

Generates the `<S>` and `</S>` tags, which direct the browser to render the text they surround in strikethrough type.

Parameters

`ctext` - the text to render in strikethrough type.

`cattributes` - other attributes to be included as-is in the tag.

Generates

`<S cattributes>ctext</S>`

http.sample

Syntax

```
http.sample (
    ctext      in      varchar2
    cattributes      in      varchar2  DEFAULT NULL);
htf.sample (ctext, cattributes) return varchar2;
```

Purpose

Generates the <SAMP> and </SAMP> tags, which direct the browser to render the text they surround in monospace font.

Parameters

ctext - the text to render in monospace font.

cattributes - other attributes to be included as-is in the tag.

Generates

<SAMP cattributes>ctext</SAMP>

htp.script

Syntax

```
htp.script(
    cscript      in      varchar2
    clangauge   in      varchar2  DEFAULT NULL);
htf.script(cscript, clangauge) return varchar2;
```

Purpose

Generates the <SCRIPT> and </SCRIPT> tags, which contain a script written in languages such as JavaScript and VBscript.

Parameters

cscript - the text of the script. This is the text that makes up the script itself, not the name of a file containing the script.

clangauge - the language in which the script is written. If this parameter is omitted, the user's browser determines the scripting language.

Generates

```
<SCRIPT LANGUAGE=clangauge>cscript</SCRIPT>
```

Example

```
htp.script ('Erupting_Volcano', 'Javascript');
```

generates:

```
<SCRIPT LANGUAGE=Javascript>"script text here"</SCRIPT>
```

This would cause the browser to run the script enclosed in the tags.

http.small

Syntax

```
http.small(
    ctext      in      varchar2
    cattributes  in      varchar2  DEFAULT NULL);
htf.small(ctext, cattributes) return varchar2;
```

Purpose

Generates the <SMALL> and </SMALL> tags, which direct the browser to render the text they surround using a small font.

Parameters

ctext - the text to render in a small font.

cattributes - other attributes to be included as-is in the tag.

Generates

<SMALL *cattributes*>*ctext*</SMALL>

htp.strike

Syntax

```
htp.strike(  
    ctext          in      varchar2  
    cattributes   in      varchar2  DEFAULT NULL);  
htf.strike(ctext, cattributes) return varchar2;
```

Purpose

Generates the <STRIKE> and </STRIKE> tags, which direct the browser to render the text they surround in strikethrough type.

Parameters

ctext - the text to be rendered in strikethrough type.

cattributes - other attributes to be included as-is in the tag.

Generates

<STRIKE *cattributes*>*ctext*</STRIKE>

http.strong

Syntax

```
http.strong (
    ctext      in      varchar2
    cattributes      in      varchar2  DEFAULT NULL);
htf.strong (ctext, cattributes) return varchar2;
```

Purpose

Generates the and tags, which direct the browser to render the text they surround in bold.

Parameters

ctext - the text to be emphasized.

cattributes - other attributes to be included as-is in the tag.

Generates

<STRONG *cattributes*>*ctext*

htp.style

Syntax

```
htp.style(cstyle in varchar2);
htf.style(cstyle) return varchar2;
```

Purpose

Generates the <STYLE> and </STYLE> tags, which include a style sheet in your Web page. Style sheets are a feature of HTML 3.2. You can get more information about style sheets at <http://www.w3.org>.

This feature is generally not compatible with browsers that support only HTML versions 2.0 or earlier. Such browsers will ignore this tag.

Parameters

cstyle - the style information to include.

Generates

```
<STYLE>cstyle</STYLE>
```

http.sub

Syntax

```
http.sub(
    ctext          in      varchar2
    calign        in      varchar2  DEFAULT NULL
    cattributes   in      varchar2  DEFAULT NULL);
htf.sub(ctext, calign, cattributes) return varchar2;
```

Purpose

Generates the _{and} tags, which direct the browser to render the text they surround as subscript.

Parameters

- ctext - the text to render in subscript.
- calign - the value for the ALIGN attribute.
- cattributes - other attributes to be included as-is in the tag.

Generates

```
<SUB ALIGN="calign" cattributes>ctext</SUB>
```

htp.sup

Syntax

```
htp.sup(
    ctext      in      varchar2
    calign     in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.sup(ctext, calign, cattributes) return varchar2;
```

Purpose

Generates the ^{and} tags, which direct the browser to render the text they surround as superscript.

Parameters

ctext - the text to render in subscript.

calign - the value for the ALIGN attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<SUP ALIGN="calign" cattributes>ctext</SUP>
```

http.tableCaption

Syntax

```
http.tableCaption (
    ccaption      in      varchar2
    calign       in      varchar2  DEFAULT NULL
    cattributes   in      varchar2  DEFAULT NULL);
htf.tableCaption (ccaption, calign, cattributes) return varchar2;
```

Purpose

Generates the <CAPTION> and </CAPTION> tags, which place a caption in an HTML table.

Parameters

- ccaption - the text for the caption.
- calign - the value for the ALIGN attribute.
- cattributes - other attributes to be included as-is in the tag.

Generates

```
<CAPTION ALIGN="calign" cattributes>ccaption</CAPTION>
```

htp.tableData

Syntax

```
htp.tableData (
    cvalue      in      varchar2  DEFAULT NULL
    calign     in      varchar2  DEFAULT NULL
    cdp        in      varchar2  DEFAULT NULL
    cnowrap    in      varchar2  DEFAULT NULL
    rowspan    in      varchar2  DEFAULT NULL
    colspan    in      varchar2  DEFAULT NULL
    attributes  in      varchar2  DEFAULT NULL);
htf.tableData (cvalue, calign, cdp, cnowrap, rowspan, colspan,
attributes) return varchar2;
```

Purpose

Generates the <TD> and </TD> tags, which insert data into a cell of an HTML table.

Parameters

cvalue - the data for the cell in the table.

calign - the value for the ALIGN attribute.

cdp - the value for the DP attribute.

cnowrap - if the value of this parameter is not NULL, the NOWRAP attribute is added to the tag.

rowspan - the value for the ROWSPAN attribute.

colspan - the value for the COLSPAN attribute.

attributes - other attributes to be included as-is in the tag.

Generates

```
<TD ALIGN="calign" DP="cdp" ROWSPAN="rowspan" COLSPAN="colspan"  
NOWRAP attributes>cvalue</TD>
```

http.tableHeader

Syntax

```
http.tableHeader (
    cvalue      in      varchar2  DEFAULT NULL
    calign     in      varchar2  DEFAULT NULL
    cdp        in      varchar2  DEFAULT NULL
    cnowrap    in      varchar2  DEFAULT NULL
    rowspan    in      varchar2  DEFAULT NULL
    colspan    in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.tableHeader (cvalue, calign, cdp, cnowrap, rowspan, colspan,
cattributes) return varchar2;
```

Purpose

Generates the <TH> and </TH> tags, which insert a header cell in an HTML table. The <TH> tag is similar to the <TD> tag, except that the text in the rows are usually rendered in bold type.

Parameters

cvalue - the data for the cell in the table.
calign - the value for the ALIGN attribute.
cdp - the value for the DP attribute.
cnowrap - if the value of this parameter is not NULL, the NOWRAP attribute is added to the tag.
rowspan - the value for the ROWSPAN attribute.
colspan - the value for the COLSPAN attribute.
cattributes - other attributes to be included as-is in the tag.

Generates

```
<TH ALIGN="calign" DP="cdp" ROWSPAN="rowspan" COLSPAN="colspan"  
NOWRAP cattributes>cvalue</TH>
```

http.tableOpen, http.tableClose

Syntax

```
http.tableOpen (
    cborder      in      varchar2  DEFAULT NULL
    calign       in      varchar2  DEFAULT NULL
    cnowrap      in      varchar2  DEFAULT NULL
    cclear       in      varchar2  DEFAULT NULL
    cattributes   in      varchar2  DEFAULT NULL);
htf.tableOpen (cborder, calign, cnowrap, cclear, cattributes) return
varchar2;

http.tableClose;
htf.tableClose return varchar2;
```

Purpose

Generates the <TABLE> and </TABLE> tags, which define an HTML table.

Parameters

cborder - the value for the BORDER attribute.

calign - the value for the ALIGN attribute.

cnowrap - if the value of this parameter is not NULL, the NOWRAP attribute is added to the tag.

cclear - the value for the CLEAR attribute.

cattributes - other attributes to be included as-is in the tag.

Generates

```
<TABLE "cborder" NOWRAP ALIGN="calign" CLEAR="cclear" cattributes>
</TABLE>
```

http.tableRowOpen, http.tableRowClose

Syntax

```
http.tableRowOpen (
    calign      in      varchar2  DEFAULT NULL
    valign      in      varchar2  DEFAULT NULL
    cdp         in      varchar2  DEFAULT NULL
    cnowrap     in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL);
htf.tableRowOpen (calign, valign, cdp, cnowrap, cattributes) return
varchar2;

http.tableRowClose;
http.tableRowClose return varchar2;
```

Purpose

Generates the <TR> and </TR> tags, which inserts a new row in an HTML table.

Parameters

calign - the value for the ALIGN attribute.
valign - the value for the VALIGN attribute.
cdp - the value for the DP attribute.
cnowrap - if the value of this parameter is not NULL, the NOWRAP attribute is added to the tag.
cattributes - other attributes to be included as-is in the tag.

Generates

```
<TR ALIGN="calign" VALIGN="valign" DP="cdp" NOWRAP cattributes>
</TR>
```

htp.teletype

Syntax

```
htp.teletype (
    ctext          in      varchar2
    cattributes   in      varchar2  DEFAULT NULL);
htf.teletype (ctext, cattributes) return varchar2;
```

Purpose

Generates the <TT> and </TT> tags, which direct the browser to render the text they surround in a fixed width typewriter font, for example, the courier font.

Parameters

ctext - the text to render in a fixed width typewriter font.

cattributes - other attributes to be included as-is in the tag.

Generates

<TT *cattributes*>*ctext*</TT>

http.title

Syntax

```
http.title (ctitle in varchar2);
htf.title (ctitle) return varchar2;
```

Purpose

Generates the <TITLE> and </TITLE> tags, which specify the text to display in the titlebar of the browser window.

Parameters

ctitle - the text to display in the titlebar of the browser window.

Generates

```
<TITLE>ctitle</TITLE>
```

htp.ulistOpen, htp.ulistClose

Syntax

```
htp.ulistOpen (
    cclear      in      varchar2  DEFAULT NULL
    cwrap       in      varchar2  DEFAULT NULL
    cdingbat   in      varchar2  DEFAULT NULL
    csrc        in      varchar2  DEFAULT NULL
    cattributes in      varchar2  DEFAULT NULL
    htf.ulistOpen (cclear, cwrap, cdingbat, csrc, cattributes) return
    varchar2;

    htp.ulistClose;
    htf.ulistClose return varhar2;
```

Purpose

Generates the and tags, which define an unordered list. An unordered list presents listed items marked off by bullets. You add list items with [htp.listItem](#).

Parameters

cclear - the value for the CLEAR attribute.
cwrap - the value for the WRAP attribute.
cdingbat - the value for the DINGBAT attribute.
csrc - the value for the SRC attribute.
cattributes - other attributes to be included as-is in the tag.

Generates

```
<UL CLEAR="cclear" WRAP="cwrap" DINGBAT="cdingbat" SRC="csrc" cat-
tributes>

</UL>
```

http.underline

Syntax

```
http.underline(  
    ctext      in      varchar2  
    cattributes  in      varchar2  DEFAULT NULL);  
htf.underline(ctext, cattributes) return varchar2;
```

Purpose

Generates the <U> and </U> tags, which direct the browser to render the text they surround with an underline.

Parameters

ctext - the text to render with an underline.

cattributes - other attributes to be included as-is in the tag.

Generates

<U *cattributes*>*ctext*</U>

htp.variable

Syntax

```
htp.variable (
    ctext          in      varchar2
    cattributes   in      varchar2  DEFAULT NULL);
htf.variable (ctext, cattributes) return varchar2;
```

Purpose

Generates the <VAR> and </VAR> tags, which direct the browser to render the text they surround in italics.

Parameters

ctext - the text to render in italics.

cattributes - other attributes to be included as-is in the tag.

Generates

<VAR *cattributes*>*ctext*</VAR>

http.wbr

Syntax

```
http.wbr;  
htf.wbr return wbr;
```

Purpose

Generates the <WBR> tag, which inserts a soft line break within a section of NOBR text.

Parameters

None.

Generates

<WBR>

The owa_content Package

Use the owa_content package to query the content service repository and manipulate document properties.

Summary

[owa_content.bigstring_arr](#) data type - lists an array of attribute values
[owa_content.delete_attribute procedure](#) - deletes the document attributes
[owa_content.delete_document procedure](#) - deletes several documents
[owa_content.delete_documents procedure](#) - deletes a document
[owa_content.document_exists function](#) - verifies if a document exists
[owa_content.get_attribute procedure](#) - retrieves attribute information
[owa_content.get_attributes function](#) - retrieves information on several attributes
[owa_content.get_author function](#) - retrieves an author of a document
[owa_content.get_char_set function](#) - retrieves a character set
[owa_content.get_content_type function](#) - retrieves content type of a document
[owa_content.get_description function](#) - retrieves a document description
[owa_content.get_encoding function](#) - retrieves the encoding of a document
[owa_content.get_expiration function](#) - retrieves a document expiration date
[owa_content.get_language function](#) - retrieves a language
[owa_content.get_length function](#) - retrieves the length of a document
[owa_content.get_title function](#) - retrieves the title of a document

[**owa_content.list_attributes procedure**](#) - lists document attributes
[**owa_content.list_documents function**](#) - lists the fullname of documents
[**owa_content.list_system_attributes procedure**](#) - lists system attributes
[**owa_content.list_user_attributes procedure**](#) - lists user attributes
[**owa_content.number_arr data type**](#) - lists attribute types
[**owa_content.rename_document procedure**](#) - renames a document
[**owa_content.set_attribute procedure**](#) - sets an attribute
[**owa_content.set_author procedure**](#) - sets an author of a document
[**owa_content.set_char_set procedure**](#) - sets a character
[**owa_content.set_content_type procedure**](#) - sets the content type of a document
[**owa_content.set_description procedure**](#) - sets a document description
[**owa_content.set_encoding procedure**](#) - sets the encoding of a document
[**owa_content.set_expiration procedure**](#) - sets a document expiration date
[**owa_content.set_language procedure**](#) - sets a language
[**owa_content.set_title procedure**](#) - sets the title of a document
[**owa_content.string_arr data type**](#) - lists the document and attribute names

owa_content.bigstring_arr data type

Definition

```
type bigstring_arr is table of varchar2 (2000) index by binary_integer;
```

Purpose

Use this data type to get an array of attribute values.

Return Value

Not applicable.

owa_content.delete_attribute procedure

Syntax

```
owa_content.delete_attribute(  
    doc_fullname    in      varchar2,  
    attrib_name     in      varchar2);
```

Purpose

Use this procedure to delete a user-defined attribute for a document.

Parameters

`doc_fullname` - the document name.
`attrib_name` - the attribute name.

Return Value

None, but if a document does not exist, a `NO_DATA_FOUND` exception is raised.

See Also

[owa_content.get_attribute procedure](#), [owa_content.set_attribute procedure](#).

owa_content.delete_document procedure

Syntax

```
owa_content.delete_document(doc_fullname in varchar2);
```

Purpose

Use this procedure to delete a document.

Parameters

doc_fullname - the document name.

Return Value

None, but if a document does not exist, a NO_DATA_FOUND exception is raised.

owa_content.delete_documents procedure

Syntax

```
owa_content.delete_documents(doc_fullname_list in string_arr);
```

Purpose

Use this procedure to delete documents that are in the `doc_fullname` list.

Parameters

`doc_fullname_list` - the document list of fullnames.

Return Value

None, but a `NO_DATA_FOUND` exception is raised if files do not exist. The delete operation stops at the same element, so make sure all files exist before using this procedure.

owa_content.document_exists function

Syntax

```
owa_content.document_exists(doc_fullname in varchar2) return boolean;
```

Purpose

Use this function to check whether the document name within the doc_fullname exists.

Parameters

doc_fullname - the document name.

Return Value

Return true if document exists.

owa_content.get_attribute procedure

Syntax

```
owa_content.get_attribute(
    doc_fullname    in      varchar2,
    attrib_name     in      varchar2,
    attrib_type     out     number,
    attrib_val      in out  varchar2);
```

Purpose

Use this procedure to get information about an attribute.

Parameters

- doc_fullname - the document name.
- attrib_name - the attribute name.
- attrib_type - the attribute type.
- attrib_val - the attribute value.

Return Value

None.

Usage

You can use the owa_content_get_attribute procedure to get information about the following predefined attributes:

Table 2–1 Predefined attributes

Attribute Name	Attribute Type	Description
author	WRBPT_STRING	The name of the document author.
method	WRBPT_STRING	Determine whether the document is stored in a database or a file system.
content_type	WRPT_STRING	The MIME type and subtype of the document.
method_info	WRBPT_STRING	The file system path of a document stored in a file system. This attribute is NULL for documents stored in databases.
creation_date	WRBPT_DATE	The date the document was created.
name	WRBPT_STRING	The document name.
description	WRBPT_STRING	A text string describing the document. This string must be no longer than 2000 bytes.
encoding	WRBPT_STRING	The encoding applied to the document, if any, such as the compress or gzip.
owner	WRBPT_STRING	The user ID of the document owner.
expires	WRBPT_DATE	The date after which you may delete the document.
path	WRBPT_STRING	The full path of the document in the content repository, which is the concatenation of the folder and name attributes.
folder	WRBPT_STRING	The content repository folder that contains the document.
title	WRBPT_STRING	The HTML title of the document, if any.

Table 2–1 Predefined attributes

Attribute Name	Attribute Type	Description
language	WRBPT_STRING	The language in which the document is written.
type	WRBPT_TYPE	Whether the document is in text or binary format.

owa_content.get_attributes function

Syntax

```
owa_content.get_attributes(
    doc_fullname    in      varchar2,
    attrib_name     out     string_arr,
    attrib_type     out     number_arr,
    attrib_val      out     bigstring_arr)
return number;
```

Purpose

Use this function to get information about all attributes for a document. Predefined attributes proceed user-defined attributes in the output tables.

Parameters

doc_fullname - the document name.
attrib_name - the attribute name.
attrib_type - the attribute type.
attrib_val - the attribute value.

Return Value

The number of attributes is given and a NO_DATA_FOUND exception is raised if the document does not exist.

See Also

[owa_content.get_attribute procedure](#), [owa_content.set_attribute procedure](#).

owa_content.get_author function

Syntax

```
owa_content.get_author(doc_fullname in varchar2) return varchar2;
```

Purpose

Use this function to get the author of a document.

Parameters

`doc_fullname` - the document name.

Return Value

A document author value is returned and a `NO_DATA_FOUND` exception is raised when a document does not exist.

owa_content.get_char_set function

Syntax

```
owa_content.char_set(doc_fullname in varchar2) return varchar2;
```

Purpose

Use this function to get the character set for a document.

Parameters

doc_fullname - the document name.

Return Value

A character set value is returned and a NO_DATA_FOUND exception is raised when a document does not exist.

owa_content.get_content_type function

Syntax

```
owa_content.get_content_type(doc_fullname in varchar2) return varchar2;
```

Purpose

Use this function to get the content type of a document.

Parameters

`doc_fullname` - the document name.

Return Value

A content type value is returned and a `NO_DATA_FOUND` exception is raised if the document does not exist.

owa_content.get_description function

Syntax

```
owa_content.get_description(doc_fullname in varchar2) return varchar2;
```

Purpose

Use this function to get the description of a document.

Parameters

doc_fullname - the document name.

Return Value

A description value is returned and a NO_DATA_FOUND exception is raised if the document does not exist.

owa_content.get_encoding function

Syntax

```
owa_content.get_encoding(doc_fullname in varchar2) return varchar2;
```

Purpose

Use this function to get encoding information for a document.

Parameters

`doc_fullname` - the document name.

Return Value

An encoding value is returned and a `NO_DATA_FOUND` exception is raised if the document does not exist.

owa_content.get_expiration function

Syntax

```
owa_content.get_expiration(doc_fullname in varchar2) return date;
```

Purpose

Use this function to get the expiration date of a document.

Parameters

doc_fullname - the document name.

Return Value

An expiration date value is returned and a NO_DATA_FOUND exception is raised if the document does not exist.

owa_content.get_language function

Syntax

```
owa_content.get_language(doc_fullname in varchar2) return varchar2;
```

Purpose

Use this function to get the language of a document.

Parameters

`doc_fullname` - the document name.

Return Value

A language value is returned and a `NO_DATA_FOUND` exception is raised if the document does not exist.

owa_content.get_length function

Syntax

```
owa_content.get_length(doc_fullname in varchar2) return number;
```

Purpose

Use this function to get the length of a document.

Parameters

doc_fullname - the document name.

Return Value

The size of the document is returned and a NO_DATA_FOUND exception is raised if the document does not exist.

owa_content.get_title function

Syntax

```
owa_content.get_title(doc_fullname in varchar2) return varchar2;
```

Purpose

Use this function to get the title of a document.

Parameters

`doc_fullname` - the document name.

Return Value

The document title is returned and a `NO_DATA_FOUND` exception is raised if the document does not exist.

owa_content.list_attributes procedure

Syntax

```
owa_content.list_attributes(  
    doc_fullname    in      varchar2,  
    attrib_name     out      string_arr);
```

Purpose

Use this procedure to list all attribute names for a document.

Parameters

doc_fullname - the document name.

attrib_name - the attribute name.

Return Value

None, but a NO_DATA_FOUND exception is raised if the document does not exist.

See Also

[owa_content.get_attribute procedure](#), [owa_content.set_attribute procedure](#).

owa_content.list_documents function

Syntax

```
owa_content.list_documents(  
    doc_list      out      string_arr)  
return binary_integer;
```

Purpose

Use this function to get a listing of documents.

Parameters

`doc_list` - the document name.

Return Value

The number of documents in the `doc_list` is returned.

owa_content.list_system_attributes procedure

Syntax

```
owa_content.list_system_attributes(  
    doc_fullname    in      varchar2,  
    attrib_name     out      string_arr);
```

Purpose

Use this procedure to get a listing of the pre-defined attributes for a document.

Parameters

doc_fullname - the document name.
attrib_name - lists the attribute names.

Return Value

None, but a NO_DATA_FOUND exception is raised if the document does not exist.

owa_content.list_user_attributes procedure

Syntax

```
owa_content.list_user_attributes(  
    doc_fullname    in      varchar2,  
    attrib_name     out      string_arr);
```

Purpose

Use this procedure to get a listing of the user-defined attributes for a document.

Parameters

`doc_fullname` - the document name.
`attrib_name` - lists the attribute names.

Return Value

None, but a `NO_DATA_FOUND` exception is raised if the document does not exist.

`owa_content.number_arr` data type

Definition

`number_arr` is table of `number` index by `binary_integer`;

Purpose

This data type can be used to get a list of attribute types.

Return Value

Not applicable.

owa_content.rename_document procedure

Syntax

```
owa_content.rename_document(
    old_fullname    in      varchar2,
    new_fullname   in      varchar2);
```

Purpose

Use this procedure to rename a document.

Parameters

`old_fullname` - the original document name.
`new_fullname` - the new document fullname.

Return Value

A `NO_DATA_FOUND` exception is raised if the `old_fullname` does not exist.

owa_content.set_attribute procedure

Syntax

```
owa_content.set_attribute(  
    doc_fullname    in      varchar2,  
    attrib_name     in      varchar2,  
    attrib_type     in      number,  
    attrib_val      in      varchar2);
```

Purpose

Use this procedure to set an attribute.

Parameters

- doc_fullname - the document name.
- attrib_name - the attribute name.
- attrib_type - the attribute type.
- attrib_val - the attribute value.

Return Value

None, but a NO_DATA_FOUND exception is raised if the document does not exist.

Usage

You cannot set the following read-only attributes: creation_date, last_modified, oid, owner, and path.

See Also

[owa_content.get_attribute procedure](#).

owa_content.set_author procedure

Syntax

```
owa_set_author(
    doc_fullname    in      varchar2,
    author_val      in      varchar2);
```

Purpose

Use this procedure to set an author of a document.

Parameters

`doc_fullname` - the document name.
`author_val` - the author name.

Return Value

None, but a `NO_DATA_FOUND` exception is raised if the document does not exist.

owa_content.set_char_set procedure

Syntax

```
owa_set_char_set(  
    doc_fullname    in      varchar2,  
    char_set_val   in      varchar2);
```

Purpose

Use this procedure to set a document character set.

Parameters

doc_fullname - the document name.

char_set_val - the character set value.

Return Value

None, but a NO_DATA_FOUND exception is raised if the document does not exist.

owa_content.set_content_type procedure

Syntax

```
owa_set_content_type(
    doc_fullname    in      varchar2,
    content_type_val in      varchar2);
```

Purpose

Use this procedure to set a document content type.

Parameters

`doc_fullname` - the document name.
`content_type_val` - the character set value.

Return Value

None, but a `NO_DATA_FOUND` exception is raised if the document does not exist.

owa_content.set_description procedure

Syntax

```
owa_set_description(  
    doc_fullname    in      varchar2,  
    desc_val        in      varchar2);
```

Purpose

Use this procedure to set a document description.

Parameters

doc_fullname - the document name.

desc_val - the description value.

Return Value

None, but a NO_DATA_FOUND exception is raised if the document does not exist.

owa_content.set_encoding procedure

Syntax

```
owa_set_encoding(
    doc_fullname    in      varchar2,
    encode_val      in      varchar2);
```

Purpose

Use this procedure to set encoding for a document.

Parameters

`doc_fullname` - the document name.
`encode_val`- the encoding value.

Return Value

None, but a `NO_DATA_FOUND` exception is raised if the document does not exist.

owa_content.set_expiration procedure

Syntax

```
owa_set_expiration(  
    doc_fullname    in      varchar2,  
    expire_val     in      date);
```

Purpose

Use this procedure to set the expiration date of a document.

Parameters

doc_fullname - the name of document that is to expire.

expire_val - the expiration date.

Return Value

None, but a NO_DATA_FOUND exception is raised if the document does not exist.

owa_content.set_language procedure

Syntax

```
owa_set_expiration(
    doc_fullname    in      varchar2,
    language_val   in      varchar2);
```

Purpose

Use this procedure to set the language of a document.

Parameters

`doc_fullname` - the document name.
`language_val` - the language type.

Return Value

None, but a `NO_DATA_FOUND` exception is raised if the document does not exist.

owa_content.set_title procedure

Syntax

```
owa_set_expiration(  
    doc_fullname    in      varchar2,  
    title_val       in      varchar2);
```

Purpose

Use this procedure to set a title for a document.

Parameters

doc_fullname - the document name.

title_val - the title name.

Return Value

None, but a NO_DATA_FOUND exception is raised if the document does not exist.

owa_content.string_arr data type

Definition

`string_arr` is table of `varchar2 (256)` index by `binary_integer`;

Purpose

Use this data type to list document and attribute names.

Return Value

Not applicable.

3

The owa_cookie Package

The owa_cookie package enables you to send and retrieve cookies in HTTP headers. For information about cookies, see these sites:

- http://home.netscape.com/newsref/std/cookie_spec.html
- <http://www.virtual.net/Projects/Cookies/>

Summary

[owa_cookie.cookie data type](#) - data type to contain cookie name-value pairs

[owa_cookie.get function](#) - gets the value of the specified cookie

[owa_cookie.get_all procedure](#) - gets all cookie name-value pairs

[owa_cookie.remove procedure](#) - removes the specified cookie

[owa_cookie.send procedure](#) - generates a “Set-Cookie” line in the HTTP header

Note: All HTTP headers have to be in English. If the headers are being generated from the database, make sure they are created in the English language.

owa_cookie.cookie data type

Definition

```
type cookie is RECORD (
    name          varchar2(4000),
    vals          vc_arr,
    num_vals      integer);
```

Description

Since the HTTP standard allows cookie names to be overloaded (that is, multiple values can be associated with the same cookie name), this is a PL/SQL RECORD holding all values associated with a given cookie name.

Type vc_arr is table of varchar2(4000) index by binary_integer.

Return Value

Not applicable.

owa_cookie.get function

Syntax

```
owa_cookie.get(name in varchar2) return cookie;
```

Purpose

This function returns the values associated with the specified cookie. The values are returned in a owa_cookie.cookie data type.

Parameters

name - the name of the cookie.

Return Value

An [owa_cookie.cookie data type](#).

owa_cookie.get_all procedure

Syntax

```
owa_cookie.get_all(
    names          out      vc_arr,
    vals           out      vc_arr,
    num_vals       out      integer);
```

Purpose

This procedure returns all cookie names and their values from the client's browser. The values appear in the order in which they were sent from the browser.

Parameters

names - the names of the cookies.
vals - the values of the cookies.
num_vals - the number of cookie-value pairs.

Generates

Arrays of the names and values in the order received, and the count of the combinations.

owa_cookie.remove procedure

Syntax

```
owa_cookie.remove(  
    name      in      varchar2,  
    val       in      varchar2,  
    path      in      varchar2  DEFAULT NULL);
```

Purpose

This procedure forces a cookie to expire immediately by setting the “expires” field of a Set-Cookie line in the HTTP header to “01-Jan-1990”. This procedure must be called within the context of an HTTP header.

Parameters

- name - the name of the cookie to expire.
- value - the value of the cookie.
- path - currently unused.

Generates

Set-Cookie: <name>=<value> expires=01-JAN-1990

owa_cookie.send procedure

Syntax

```
owa_cookie.send(  
    name          in      varchar2,  
    value         in      varchar2,  
    expires       in      date      DEFAULT NULL,  
    path          in      varchar2 DEFAULT NULL,  
    domain        in      varchar2 DEFAULT NULL,  
    secure        in      varchar2 DEFAULT NULL);
```

Purpose

This procedure generates a Set-Cookie line, which transmits a cookie to the client. This procedure must occur in the context of an HTTP header.

Parameters

- name -the name of the cookie.
- value - the value of the cookie.
- expires - the date at which the cookie will expire.
- path - the value for the path field.
- domain - the value for the domain field.
- secure - if the value of this parameter is not NULL, the “secure” field is added to the line.

Generates

Set-Cookie: <name>=<value> expires=<expires> path=<path> domain=<domain>
secure

The owa_image Package

The owa_image package enables you to create an imagemap that invokes a PL/SQL cartridge when clicked.

Summary

[owa_image.NULL_POINT package variable](#) - variable of type point whose X and Y values are NULL

[owa_image.point data type](#) - data type to contain the X and Y values of a coordinate

[owa_image.get_x function](#) - gets the X value of a point type

[owa_image.get_y function](#) - gets the Y value of a point type

owa_image.NULL_POINT package variable

Syntax

null_point - package variable

Definition

This package variable of type `point` is used to default point parameters. Both the X and the Y fields of this variable are NULL.

Return Value

Not applicable.

owa_image.point data type

Syntax

point - data type

Definition

This data type provides the x and y coordinates of a user's click on an imagemap. It is defined as:

type point is table of varchar2(32767) index by binary_integer.

Return Value

Not applicable.

owa_image.get_x function

Syntax

```
owa_image.get_x(p in point) return integer;
```

Purpose

This function returns the X coordinate of the point where the user clicked on an image map.

Parameters

p - the point where the user clicked.

Return Value

The X coordinate as an integer.

owa_image.get_y function

Syntax

```
owa_image.get_y(p in point) return integer;
```

Purpose

This function returns the Y coordinate of the point where the user clicked on an image map.

Parameters

p - the point where the user clicked.

Return Value

The Y coordinate as an integer.

`owa_image.get_y` function

The `owa_opt_lock` Package

The `owa_opt_lock` package implements an optimistic locking scheme that you can use in your stored procedure to check if the row that the user is interested in updating has been changed by someone else in the meantime.

Summary

[`owa_opt_lock_vcArray` data type](#) - data type to contain ROWIDs

[`owa_opt_lock.checksum` function](#) - returns the checksum value

[`owa_opt_lock.get_rowid` function](#) - returns the ROWID value

[`owa_opt_lock.store_values` procedure](#) - stores unmodified values in hidden fields for later verification

[`owa_opt_lock.verify_values` function](#) - verifies the stored values against modified values

owa_opt_lock_vcArray data type

Syntax

owa_opt_lock_vcArray - data type

Definition

This data type is a PL/SQL table intended to hold ROWIDs.

Type vcArray is table of varchar2(2000) index by binary_integer.

Note: This is different from the [owa_text_vc_arr data type](#).

Return Value

Not applicable.

owa_opt_lock.checksum function

Syntax

```
owa_opt_lock.checksum(p_buff in varchar2) return number;
owa_opt_lock.checksum(
    p_owner      in      varchar2
    p_tname      in      varchar2
    p_rowid      in      rowid) return number;
```

Purpose

This function returns a checksum value for a specified string, or for a row in a table. For a row in a table, the function calculates the checksum value based on the values of the columns in the row. This function comes in two versions.

The first version returns a checksum based on the specified string. This is a “pure” 32-bit checksum executed by the database and based on the Internet 1 protocol.

The second version returns a checksum based on the values of a row in a table. This is a “impure” 32-bit checksum based on the Internet 1 protocol.

Parameters

p_buff - the string for which you want to calculate the checksum.

p_owner - the owner of the table.

p_tname - the table name.

p_rowid - the row in *p_tname* for which you want to calculate the checksum value. You can use the [owa_opt_lock.get_rowid function](#) to convert vcArray values to proper rowids.

Return Value

A checksum value.

owa_opt_lock.get_rowid function

Syntax

```
owa_opt_lock.get_rowid(p_old_values in vcArray) return rowid;
```

Purpose

This function returns the ROWID data type from the specified [owa_opt_lock_vcArray data type](#).

Parameters

`p_old_values` - this parameter is usually passed in from an HTML form.

Return Value

A ROWID.

owa_opt_lock.store_values procedure

Syntax

```
owa_opt_lock.store_values(
    p_owner      in      varchar2
    p_tname      in      varchar2
    p_rowid      in      rowid);
```

Purpose

This procedure stores the column values of the row that you want to update later. The values are stored in hidden HTML form elements. Before you update the row, compare these values with the current row values to ensure that the values in the row have not been changed. If the values have been changed, you can warn the users and let them decide if the update should still take place.

Parameters

- p_owner - the owner of the table.
- p_tname - the name of the table.
- p_rowid - the row for which you want to store values.

Generates

A series of hidden form elements:

- One hidden form element is created for the table owner. The name of the element is “old_p_tname”, where *p_tname* is the name of the table. The value of the element is the owner name.
- One hidden form element is created for the table name. The name of the element is “old_p_tname”, where *p_tname* is the name of the table. The value of the element is the table name.
- One element is created for each column in the row. The name of the element is “old_p_tname”, where *p_tname* is the name of the table. The value of the element is the column value.

See Also

[owa_opt_lock.verify_values function.](#)

owa_opt_lock.verify_values function

Syntax

```
owa_opt_lock.verify_values(p_old_values in vcArray) return boolean;
```

Purpose

This function verifies whether values in the specified row have been updated since the last query. This function is used with the [owa_opt_lock.store_values procedure](#).

Parameters

p_old_values - a PL/SQL table containing the following information:

- p_old_values(1) specifies the owner of the table.
- p_old_values(2) specifies the table.
- p_old_values(3) specifies the rowid of the row you want to verify.
- The remaining indexes contain values for the columns in the table.

Typically, this parameter is passed in from the HTML form, where you have previously called the [owa_opt_lock.store_values procedure](#) to store the row values on hidden form elements.

Return Value

TRUE if no other update has been performed; otherwise, FALSE.

See Also

[owa_opt_lock.store_values procedure](#).

6

The owa_pattern Package

The owa_pattern package in the PL/SQL Web Toolkit enables you to locate text patterns within strings and replace the matched string with another string. You can use regular expressions with the subprograms in this package.

Summary

[Regular Expressions](#) - this section describes the special characters, quantifiers, and flags that you can use in forming regular expressions.

[owa_pattern.amatch function](#) - determines if a string contains the specified pattern

[owa_pattern.change function and procedure](#) - replaces a pattern within a string

[owa_pattern.getpat procedure](#) - generates a pattern data type from a VARCHAR2 type

[owa_pattern.match function](#) - determines if a string contains the specified pattern

[owa_pattern.pattern data type](#) - data type used to store regular expressions

Regular Expressions

You specify a regular expression by creating the string you want to match interspersed with various wildcard tokens and quantifiers.

Wildcard Tokens

Wildcard tokens match something other than themselves:

Table 6–1 *Wildcard tokens recognized by owa_pattern package*

Token	Description
^	Matches newline or the beginning of the target
\$	Matches newline or the end of the target
\n	Matches newline
.	Matches any character except newline
\t	Matches tab
\d	Matches digits [0-9]
\D	Matches non-digits [not 0-9]
\w	Matches word characters (0-9, a-z, A-Z, or _)
\W	Matches non-word characters (not 0-9, a-z, A-Z, or _)
\s	Matches whitespace characters (blank, tab, or newline).
\S	Matches non-whitespace characters (not blank, tab, or newline)
\b	Matches “word” boundaries (between \w and \W)
\x<HEX>	Matches the value in the current character set of the two hexadecimal digits
\<OCT>	Matches the value in the current character set of the two or three octal digits
\	Followed by any character not covered by another case matches that character
&	Applies only to CHANGE. This causes the string that matched the regular expression to be included in the string that replaces it. This differs from the other tokens in that it specifies how a target is changed rather than how it is matched. This is explained further under CHANGE.

Quantifiers

Any tokens except & can have their meaning extended by any of the following quantifiers. You can also apply these quantifiers to literals:

Table 6–2 Quantifiers

Quantifier	Description
?	0 or 1 occurrence(s)
*	0 or more occurrences
+	1 or more occurrence(s)
{n}	Exactly <i>n</i> occurrences
(n,)	At least <i>n</i> occurrences
{n, m}	At least <i>n</i> , but not more than <i>m</i> , occurrences

Flags

In addition to targets and regular expressions, the owa_pattern functions and procedures can use flags to affect how they are interpreted.

Table 6–3 Flags

Flag	Description
i	This indicates a case-insensitive search.
g	This applies only to CHANGE. It indicates a global replace. That is, all portions of the target that match the regular expression are replaced.

owa_pattern.amatch function

Syntax

```
owa_pattern.amatch(
    line      in      varchar2
    from_loc in      integer
    pat       in      varchar2
    flags     in      varchar2  DEFAULT NULL) return integer;

owa_pattern.amatch(
    line      in      varchar2
    from_loc in      integer
    pat       in out   pattern
    flags     in      varchar2  DEFAULT NULL) return integer;

owa_pattern.amatch(
    line      in      varchar2
    from_loc in      integer
    pat       in      varchar2
    backrefs out     owa_text_vc_arr
    flags     in      varchar2  DEFAULT NULL) return integer;

owa_pattern.amatch(
    line      in      varchar2
    from_loc in      integer
    pat       in out   pattern
    backrefs out     owa_text_vc_arr
    flags     in      varchar2  DEFAULT NULL) return integer;
```

Purpose

This function enables you to specify if a pattern occurs in a particular location in a string. There are four versions to this function:

- The first and second versions of the function do not save the matched tokens (these are saved in the *backrefs* parameters in the third and fourth versions). The difference between the first and second versions is the *pat* parameter, which can be a VARCHAR2 or a pattern data type.

- The third and fourth versions of the function save the matched tokens in the *backrefs* parameter. The difference between the third and fourth versions is the *pat* parameter, which can be a VARCHAR2 or a pattern data type.

Note: If multiple overlapping strings can match the regular expression, this function takes the longest match.

Parameters

line - the text to search in.

from_loc - the location (in number of characters) in *line* where the search is to begin.

pat - the string to match. It can contain regular expressions. This can be either a VARCHAR2 or a pattern. If it is a pattern, the output value of this parameter is the pattern matched.

backrefs - the text that is matched. Each token that is matched is placed in a cell in the [owa_text_vc_arr data type](#) PL/SQL table.

flags - whether or not the search is case-sensitive. If the value of this parameter is "i", the search is case-insensitive. Otherwise the search is case-sensitive.

Return Value

The index of the character after the end of the match, counting from the beginning of *line*. If there was no match, the function returns 0.

owa_pattern.change function and procedure

Syntax

```
/* function */
owa_pattern.change(
    line          in out  varchar2
    from_str     in      varchar2
    to_str       in      varchar2
    flags        in      varchar2  DEFAULT NULL) return integer;

/* procedure */
owa_pattern.change(
    line          in out  varchar2
    from_str     in      varchar2
    to_str       in      varchar2
    flags        in      varchar2  DEFAULT NULL);

/* function */
owa_pattern.change(
    mline         in out  owa_text.multi_line
    from_str     in      varchar2
    to_str       in      varchar2
    flags        in      varchar2  DEFAULT NULL) return integer;

/* procedure */
owa_pattern.change(
    mline         in out  owa_text.multi_line
    from_str     in      varchar2
    to_str       in      varchar2
    flags        in      varchar2  DEFAULT NULL);
```

Purpose

This function or procedure performs a search and replace on a string or multi_line data type.

Note: if multiple overlapping strings can match the regular expression, this function takes the longest match.

Parameters

line - the text to search in. The output value of this parameter is the altered string.
mline - the text to search in. This is a [owa_text.multi_line data type](#) data type. The output value of this parameter is the altered string.
from_str - the regular expression to replace.
to_str - the substitution pattern.
flags - whether or not the search is case-sensitive, and whether or not changes are to be made globally. If “i” is specified, the search is case-insensitive. If “g” is specified, changes are made to all matches. Otherwise, the function stops after the first substitution is made.

Return Value

The number of substitutions made.

Example

In the following example, *num_found* is 1, and *theline* is changed to “what is the idea?”.

```
create or replace procedure test_pattern as
  theline VARCHAR2(256);
  num_found integer;
begin
  theline := 'what is the goal?';
  num_found := owa_pattern.change(theline, 'goal', 'idea', 'g');
  htp.print(num_found); -- num_found is 1
  htp.print(theline); -- theline is 'what is the idea?'
end;
/
show errors
```

owa_pattern.getpat procedure

Syntax

```
owa_pattern.getpat(arg in VARCHAR2, pat in out pattern);
```

Purpose

This procedure converts a VARCHAR2 string into a [owa_pattern.pattern data type](#).

Parameters

arg - the string to convert.

pat - the [owa_pattern.pattern data type](#) initialized with *arg*.

Returns

None.

owa_pattern.match function

Syntax

```
owa_pattern.match(
    line      in      varchar2
    pat       in      varchar2
    flags     in      varchar2 DEFAULT NULL) return boolean;

owa_pattern.match(
    line      in      varchar2
    pat       in out   pattern
    flags     in      varchar2 DEFAULT NULL) return boolean;

owa_pattern.match(
    line      in      varchar2
    pat       in      varchar2
    backrefs  out     owa_text_vc_arr
    flags     in      varchar2 DEFAULT NULL) return boolean;

owa_pattern.match(
    line      in      varchar2
    pat       in out   pattern
    backrefs  out     owa_text_vc_arr
    flags     in      varchar2 DEFAULT NULL) return boolean;

owa_pattern.match(
    mline    in      owa_text_multi_line
    pat      in      varchar2
    rlist    out    owa_text_row_list
    flags    in      varchar2 DEFAULT NULL) return boolean;

owa_pattern.match(
    mline    in      owa_text_multi_line
    pat      in out   pattern
    rlist    out    owa_text_row_list
    flags    in      varchar2 DEFAULT NULL) return boolean;
```

Purpose

This function determines if a string contains the specified pattern. Pattern can contain regular expressions.

Note that if multiple overlapping strings can match the regular expression, this function takes the longest match.

Parameters

line - the text to search in.

mline - the text to search in. This is a [owa_text.multi_line data type](#) data type.

pat - the pattern to match. This is either a VARCHAR2 or a [owa_pattern.pattern data type](#) data type. If it is a pattern, the output value of this parameter is the pattern matched.

backrefs - the text that is matched. Each token that is matched is placed in a cell in the [owa_text_vc_arr data type](#) PL/SQL table.

rlist - an output parameter containing a list of matches.

flags - whether or not the search is case-sensitive. If the value of this parameter is "i", the search is case-insensitive. Otherwise the search is case-sensitive.

Return Value

TRUE if a match was found, FALSE otherwise.

Examples

The following example searches for the string "goal" followed by any number of characters in *sometext*. If found,

```
sometext  VARCHAR2(256);
pat        VARCHAR2(256);

sometext := 'what is the goal?'
pat := 'goal.*';
if owa.pattern.match(sometext, pat) then
    htp.print('Match found');
else
    htp.print('Match not found');
end if;
```

owa_pattern.pattern data type

Syntax

owa_pattern.pattern - data type

Definition

The advantage is that you can use a pattern as both an input and output parameter. Thus, you can pass the same regular expression to OWA_PATTERN function calls, and it only has to be parsed once.

Return Value

Not applicable.

`owa_pattern` pattern data type

The owa_sec Package

This chapter describes the functions, procedures, and data types in the `owa_sec` package in the PL/SQL Web Toolkit.

Note: The procedures and functions in the `owa_sec` package should be used only when you use [custom authentication](#).

Parameters that have default values are optional.

Summary

[owa_sec.get_client_hostname function](#) - returns the client's hostname

[owa_sec.get_client_ip function](#) - returns the client's IP address

[owa_sec.get_password function](#) - returns the password that the user entered

[owa_sec.get_user_id function](#) - returns the username that the user entered

[owa_sec.set_authorization procedure](#) - enables the PL/SQL application to use custom authentication

[owa_sec.set_protection_realm procedure](#) - defines the realm that the page is in

owa_sec.get_client_hostname function

Syntax

```
owa_sec.get_client_hostname return varchar2;
```

Purpose

This function returns the hostname of the client.

Parameters

None.

Return Value

The hostname.

owa_sec.get_client_ip function

Syntax

```
owa_sec.get_client_ip return owa_util.ip_address;
```

Purpose

This function returns the IP address of the client.

Parameters

None.

Return Value

The IP address. The [owa_util.ip_address data type](#) is a PL/SQL table where the first four elements contain the four numbers of the IP address. For example, if the IP address is 123.45.67.89 and the variable *ipaddr* is of the [owa_util.ip_address](#) data type, the variable would contain the following values:

ipaddr(1) = 123
ipaddr(2) = 45
ipaddr(3) = 67
ipaddr(4) = 89

owa_sec.get_password function

Syntax

```
owa_sec.get_password return varchar2;
```

Purpose

This function returns the password that the user used to log in.

For security reasons, this function returns a true value only when custom authentication is used. If you call this function when you are not using custom authentication, the function returns an undefined value. The reason for this is not to expose database passwords.

Parameters

None.

Return Value

The password.

owa_sec.get_user_id function

Syntax

```
owa_sec.get_user_id return varchar2;
```

Purpose

This function returns the username that the user used to log in.

Parameters

None.

Return Value

The username.

owa_sec.set_authorization procedure

Syntax

```
owa_sec.set_authorization(scheme in integer);
```

Purpose

This procedure, which is called in the initialization portion of the owa_custom package, sets the authorization scheme for a PL/SQL cartridge. This enables you to implement your own authorize function, which will be used to authorize the user before the procedure requested by the user is run. The location where you should place the authorize function depends on the scheme you selected.

Parameters

scheme - the authorization scheme. It is one of:

Table 7–1 Schemes for set_authorization

Scheme	Description
OWA_SEC.NO_CHECK	Specifies that the PL/SQL application is not to do any custom authentication. This is the default.
OWA_SEC.GLOBAL	Enables you to define an authorize function that is called for all users and all procedures. The function is owa_custom.authorize function in the “oas_public” schema.
OWA_SEC.PER_PACKAGE	Enables you to define an authorize function that is called when procedures in a package or anonymous procedures are called. If the procedures are in a package, the <i>package.authorize</i> function in the user's schema is called to authorize the user. If the procedures are not in a package, then the anonymous authorize function in the user's schema is called.

Table 7–1 Schemes for set_authorization

Scheme	Description
OWA_SEC.CUSTOM	Enables you to implement different authorize functions for each user. The function <code>owa_custom.authorize</code> in the user's schema is called to authorize the user. If the user's schema does not contain an <code>owa_custom.authorize</code> function, the PL/SQL cartridge looks for it in the “oas_public” schema.

The custom `authorize` function has the following signature:

```
function authorize return boolean;
```

If the function returns TRUE, authentication succeeded. If it returns FALSE, authentication failed.

If the `authorize` function is not defined, the cartridge returns an error and fails.

Return Value

Not applicable.

owa_sec.set_protection_realm procedure

Syntax

```
owa_sec.set_protection_realm(realm in varchar2);
```

Purpose

This procedure sets the realm of the page that is returned to the user. The user needs to enter a username and login that already exists in the realm for authorization to succeed.

Parameters

realm - the realm in which the page should belong. This string is displayed to the user.

Return Value

Not applicable.

The owa_text Package

The owa_text package contains subprograms for manipulating strings.

Summary

[owa_text.add2multi procedure](#) - adds text to an existing multi_line type
[owa_text.multi_line data type](#) - data type for holding large amounts of text
[owa_text.new_row_list](#) - creates a new row_list
[owa_text.print_multi procedure](#) - prints out the contents of a multi_list
[owa_text.print_row_list procedure](#) - prints out the contents of a row_list
[owa_text.row_list data type](#) - data type for holding data to be processed
[owa_text.stream2multi procedure](#) - converts a varchar2 to a multi_line type
[owa_text_vc_arr data type](#) - data type for holding large amounts of text

owa_text.add2multi procedure

Syntax

```
owa_text.add2multi(
    stream      in      varchar2
    mline      in out  multi_line
    continue    in      boolean  DEFAULT TRUE);
```

Purpose

This procedure adds content to an existing [owa_text.multi_line data type](#).

Parameters

stream - the text to add.

mline - the [owa_text.multi_line data type](#). The output of this parameter contains *stream*.

continue - if TRUE, the procedure appends *stream* within the previous final row (assuming it is less than 32K). If FALSE, the procedure places *stream* in a new row.

Return Value

None.

owa_text.multi_line data type

Definition

```
type multi_line is record (
    rows          vc_arr,
    num_rows      integer,
    partial_row   boolean);
```

Purpose

This data type is a PL/SQL record that is used to hold large amounts of text.

The rows field, of type [owa_text_vc_arr data type](#), contains the text data in the record.

Return Value

Not applicable.

owa_text.new_row_list

Syntax

```
/* procedure */
owa_text.new_row_list(rlist out row_list);
/* function */
owa_text.new_row_list return row_list;
```

Purpose

This function or procedure creates a new [owa_text.row_list data type](#).

The function version takes no parameters and returns a new empty row_list.

The procedure version creates the row_list data type as an output parameter.

Parameters

rlist - this is an output parameter containing the new row_list data type.

Return Value

The function version returns the new row_list data type.

owa_text.print_multi procedure

Syntax

```
owa_text.print_multi(mline in multi_line);
```

Purpose

This procedure uses [http.print](#), [http.prn](#) to print the “rows” field of the [owa_text.multi_line](#) data type.

Parameters

mline - the multi_line data type to print.

Return Value

The contents of the multi_line.

owa_text.print_row_list procedure

Syntax

```
owa_text.print_row_list(rlist in row_list);
```

Purpose

This procedure uses [http.print](#), [http.prn](#) to print the “rows” field of the [owa_text.row_list data type](#).

Parameters

rlist - the row_list data type to print.

Return Value

The contents of the row_list.

owa_text.row_list data type

Definition

```
type row_list is record (
    rows          int_arr,
    num_rows      integer);
int_arr is defined as:
type int_arr is table of integer index by binary_integer;
```

Return Value

Not applicable.

owa_text.stream2multi procedure

Syntax

```
owa_text.stream2multi(  
    stream      in      varchar2  
    mline       out     multi_line);
```

Purpose

This procedure converts a string to a multi_line data type.

Parameters

stream - the string to convert.

mline - the stream in [owa_text.multi_line data type](#) format.

Return Value

None.

owa_text_vc_arr data type

Definition

```
type vc_arr is table of varchar2(32767) index by binary_integer;
```

Purpose

This is a component of the [owa_text.multi_line data type](#).

Return Value

Not applicable.

`owa_text_vc_arr` data type

The `owa_util` Package

The `owa_util` package contains utility subprograms for performing operations such as getting the value of CGI environment variables, printing the data that is returned to the client, and printing the results of a query in an HTML table.

Summary

- `owa_util.bind_variables function` - prepares a SQL query and binds variables to it
- `owa_util.calendarprint procedure` - prints a calendar
- `owa_util.cellsprint procedure` - prints the contents of a query in an HTML table
- `owa_util.choose_date procedure` - generates HTML form elements that allow the user to select a date
- `owa_util.dateType data type` - data type to hold date information
- `owa_util.get_cgi_env function` - returns the value of the specified CGI environment variable
- `owa_util.get_owa_service_path function` - returns the full virtual path for the cartridge
- `owa_util.get_procedure function` - returns the name of the procedure that is invoked by the PL/SQL cartridge
- `owa_util.http_header_close procedure` - closes the HTTP header
- `owa_util.ident_arr data type`
- `owa_util.ip_address data type`
- `owa_util.listprint procedure` - generates a HTML form element that contains data from a query.

owa_util.mime_header procedure - generates the Content-type line in the HTTP header

owa_util.print_cgi_env procedure - generates a list of all CGI environment variables and their values

owa_util.redirect_url procedure - generates the Location line in the HTTP header

owa_util.showpage procedure - prints a page generated by the htp and htf packages in SQL*Plus

owa_util.showsource procedure - prints the source for the specified subprogram

owa_util.signature procedure - prints a line that says that the page is generated by the PL/SQL Agent

owa_util.status_line procedure - generates the Status line in the HTTP header

owa_util.tablePrint function - prints the data from a table in the database as an HTML table

owa_util.todate function - converts dateType data to the standard PL/SQL date type

owa_util.who_called_me procedure - returns information on the caller of the procedure

owa_util.bind_variables function

Syntax

```
owa_util.bind_variables(
    theQuery      in      varchar2  DEFAULT NULL
    bv1Name       in      varchar2  DEFAULT NULL
    bv1Value      in      varchar2  DEFAULT NULL
    bv2Name       in      varchar2  DEFAULT NULL
    bv2Value      in      varchar2  DEFAULT NULL
    bv3Name       in      varchar2  DEFAULT NULL
    bv3Value      in      varchar2  DEFAULT NULL
    ...
    bv25Name      in      varchar2  DEFAULT NULL
    bv25Value     in      varchar2  DEFAULT NULL) return integer;
```

Purpose

This function prepares a SQL query by binding variables to it, and stores the output in an opened cursor. You normally use this function as a parameter to a procedure to which you desire to send a dynamically generated query. You can specify up to 25 bind variables.

Parameters

- theQuery - the SQL query statement. This must be a SELECT statement.
- bv1Name - the name of the variable.
- bv2Value - the value of the variable.

Return Value

An integer identifying the opened cursor.

owa_util.calendarprint procedure

Syntax

```
owa_util.calendarprint(
    p_query      in      varchar2
    p_mf_only   in      varchar2  DEFAULT 'N');

owa_util.calendarprint(
    p_cursor     in      integer
    p_mf_only   in      varchar2  DEFAULT 'N');
```

Purpose

This procedure creates a calendar in HTML. Each date in the calendar can contain any number of hypertext links. To achieve this effect, design your query as follows:

- The first column should be a DATE. This is used to correlate the information produced by the query with the calendar output automatically generated by the procedure.
Note that the query output must be sorted on this column using ORDER BY.
- The second column contains the text, if any, you want printed for that date.
- The third column contains the destination for automatically generated links. Each item in the second column becomes a hypertext link to the destination given in this column. If this column is omitted, the items in the second column are simple text, not links.

This procedure has 2 versions. Version 1 uses a hard-coded query stored in a varchar2 string. Version 2 uses a dynamic query prepared with the [owa_util.bind_variables function](#).

Parameters

p_query - a PL/SQL query. See the description above on what the query should return.

p_cursor - a PL/SQL cursor containing the same format as *p_query*.

p_mf_only - if "N" (the default), the generated calendar includes Sunday through Saturday. Otherwise, it includes Monday through Friday only.

Generates

A calendar in the form of an HTML table with a visible border.

owa_util.cellsprint procedure

Syntax

```
owa_util.cellsprint(
    p_colCnt          in  integer
    p_resultTbl       in  vc_arr
    p_format_numbers  in  varchar2  DEFAULT NULL);

owa_util.cellsprint(
    p_theQuery        in  varchar2
    p_max_rows        in  number   DEFAULT 100
    p_format_numbers  in  varchar2  DEFAULT NULL);

owa_util.cellsprint(
    p_theCursor        in  integer
    p_max_rows        in  number   DEFAULT 100
    p_format_numbers  in  varchar2  DEFAULT NULL);

owa_util.cellsprint(
    p_theQuery        in  varchar2
    p_max_rows        in  number   DEFAULT 100
    p_format_numbers  in  varchar2  DEFAULT NULL
    p_skip_rec        in  number   DEFAULT 0
    p_more_data       out boolean);

owa_util.cellsprint(
    p_theCursor        in  integer
    p_max_rows        in  number   DEFAULT 100
    p_format_numbers  in  varchar2  DEFAULT NULL
    p_skip_rec        in  number   DEFAULT 0
    p_more_data       out boolean);

owa_util.cellsprint(
    p_theQuery        in  varchar2
    p_max_rows        in  number   DEFAULT 100
    p_format_numbers  in  varchar2  DEFAULT NULL
    p_reccnt          out number);
```

```
owa_util.cellsprint(
    p_theCursor      in  integer
    p_max_rows       in  number      DEFAULT 100
    p_format_numbers in  varchar2   DEFAULT NULL
    p_reccnt         out number);

owa_util.cellsprint(
    p_theQuery       in  varchar2
    p_max_rows       in  number      DEFAULT 100
    p_format_numbers in  varchar2   DEFAULT NULL
    p_skip_rec       in  number      DEFAULT 0
    p_more_data      out boolean
    p_reccnt         out number);

owa_util.cellsprint(
    p_theCursor      in  integer
    p_max_rows       in  number      DEFAULT 100
    p_format_numbers in  varchar2   DEFAULT NULL
    p_skip_rec       in  number      DEFAULT 0
    p_more_data      out boolean
    p_reccnt         out number);
```

Purpose

This procedure generates an HTML table from the output of a SQL query. SQL atomic data items are mapped to HTML cells and SQL rows to HTML rows. You must write the code to begin and end the HTML table.

There are nine versions of this procedure:

- The first version lets you pass the results of a query into an index table. So, you perform the query and cellsprint does the formatting. To have more control in generating an HTML table from the output of an SQL query, you can use the htf.format_cell function.
- The second and third versions display rows (up to the specified maximum) returned by the query or cursor.
- The fourth and fifth versions allow you to exclude the specified number of rows from the HTML table. You can also use the fourth and fifth versions to scroll through result sets by saving the last row seen in a hidden form element.
- The sixth through ninth versions are the same as the first four versions, except that they return a row count output parameter.

Parameters

p_colCnt - the number of columns in the table.

p_theQuery - a SQL SELECT statement.

p_theCursor - a cursor ID. This can be the return value from the [owa_util.bind_variables function](#).

p_max_rows - the maximum number of rows to print.

p_format_numbers - if the value of this parameter is not NULL, number fields are right-justified and rounded to two decimal places.

p_skip_rec - the number of rows to exclude from the HTML table.

p_more_data - TRUE if there are more rows in the query or cursor, FALSE otherwise.

p_reccnt - the number of rows that have been returned by the query. This value does not include skipped rows (if any).

p_resultTbl - the index table which will contain the result of the query. Each entry in the query will correspond to one column value.

Generates

```
<tr><td>QueryResultItem</td><td>QueryResultItem</td></tr>
<tr><td>QueryResultItem</td><td>QueryResultItem</td></tr>
```

owa_util.choose_date procedure

Syntax

```
owa_util.choose_date(
    p_name      in      varchar2,
    p_date      in      date      DEFAULT SYSDATE);
```

Purpose

This procedure generates three HTML form elements that allow the user to select the day, the month, and the year.

The parameter in the procedure that receives the data from these elements should be a [owa_util.dateType data type](#). You can use the [owa_util.todate function](#) to convert the [owa_util.dateType data type](#) value to the standard Oracle7 DATE data type.

Parameters

p_name - the name of the form elements.

p_date - the initial date that is selected when the HTML page is displayed.

Generates

```
<SELECT NAME="p_name" SIZE="1">
<OPTION value="01">1
...
<OPTION value="31">31
</SELECT>
-
<SELECT NAME="p_name" SIZE="1">
<OPTION value="01">JAN
...
<OPTION value="12">DEC
</SELECT>
-
<SELECT NAME="p_name" SIZE="1">
<OPTION value="1992">1992
...
<OPTION value="2002">2002
</SELECT>
```

owa_util.dateType data type

Definition

```
type dateType is table of varchar2(10) index by binary_integer;
```

Function

The `owa_util.todate function` converts an item of this type to the type DATE, which is understood and properly handled as data by the database. The procedure `owa_util.choose_date procedure` enables the user to select the desired date.

Return Value

Not applicable.

owa_util.get_cgi_env function

Syntax

```
owa_util.get_cgi_env(param_name in varchar2) return varchar2;
```

Purpose

This function returns the value of the specified CGI environment variable. Although the WRB is not operated through CGI, many WRB cartridges, including the PL/SQL cartridge, can make use of CGI environment variables.

Parameters

param_name - the name of the CGI environment variable. It is case-insensitive.

Note: You can get the values of all CGI environment variables except for QUERY_STRING because the PL/SQL cartridge parses the value of QUERY_STRING to determine the parameters to pass to the stored procedure.

Return Value

The value of the specified CGI environment variable. If the variable is not defined, the function returns NULL.

owa_util.get_owa_service_path function

Syntax

```
owa_util.get_owa_service_path return varchar2;
```

Purpose

This function returns the full virtual path of the PL/SQL cartridge that is handling the request.

Parameters

None.

Return Value

A virtual path of the PL/SQL cartridge that is handling the request.

owa_util.get_procedure function

Syntax

```
owa_util.get_procedure return varchar2;
```

Purpose

This function returns the name of the procedure that is being invoked by the PL/SQL cartridge.

Parameters

None.

Return Value

The name of a procedure, including the package name if the procedure is defined in a package.

owa_util.http_header_close procedure

Syntax

```
owa_util.http_header_close;
```

Purpose

This procedure generates a newline character to close the HTTP header.

Use this procedure if you have not explicitly closed the header by using the `bclose_header` parameter in calls such as [owa_util.mime_header procedure](#), [owa_util.redirect_url procedure](#), or [owa_util.status_line procedure](#). The HTTP header must be closed before any `htp.print` or `htp.prn` calls.

Parameters

None.

Generates

A newline character, which closes the HTTP header.

owa_util.ident_arr data type

Definition

```
type ident_arr is table of varchar2(30) index by binary_integer;
```

Return Value

Not applicable.

owa_util.ip_address data type

Definition

```
type ip_address is table of integer index by binary_integer;
```

Purpose

This data type is used by the [owa_sec.get_client_ip function](#).

Return Value

Not applicable.

owa_util.listprint procedure

Syntax

```
owa_util.listprint(
    p_theQuery      in      varchar2
    p_cname         in      varchar2
    p_nsize         in      number
    p_multiple      in      boolean  DEFAULT FALSE);

owa_util.listprint(
    p_theCursor     in      integer
    p_cname         in      varchar2
    p_nsize         in      number
    p_multiple      in      boolean  DEFAULT FALSE);
```

Purpose

This procedure generates an HTML selection list form element from the output of a SQL query. The columns in the output of the query are handled in the following manner:

- The first column specifies the values that are sent back. These values are used for the VALUE attribute of the OPTION tag.
- The second column specifies the values that the user sees.
- The third column specifies whether or not the row is marked as SELECTED in the OPTION tag. If the value is not NULL, the row is selected.

There are two versions of this procedure. The first version contains a hard-coded SQL query, and the second version uses a dynamic query prepared with the [owa_util.bind_variables function](#).

Parameters

p_theQuery - the SQL query.

p_theCursor - the cursor ID. This can be the return value from the [owa_util.bind_variables function](#).

p_cname - the name of the HTML form element.

p_nsize - the size of the form element (this controls how many items the user can see without scrolling).

p_multiple - whether multiple selection is permitted.

Generates

```
<SELECT NAME="p_cname" SIZE="p_nsize">
<OPTION SELECTED
value='value_from_the_first_column'>value_from_the_second_column
<OPTION SELECTED
value='value_from_the_first_column'>value_from_the_second_column
...
</SELECT>
```

owa_util.mime_header procedure

Syntax

```
owa_util.mime_header(
    ccontent_type    in      varchar2 DEFAULT 'text/html',
    bclose_header   in      boolean  DEFAULT TRUE,
    ccharset        in      varchar2 DEFAULT NULL);
```

Purpose

This procedure changes the default MIME header that the cartridge returns.

This procedure must come before any `http.print` or `http.prn` calls in order to direct the cartridge not to use the default MIME header.

Parameters

`ccontent_type` - the MIME type to generate.

`bclose_header` - whether or not to close the HTTP header. If TRUE, two newlines are sent, which closes the HTTP header. Otherwise, one newline is sent, and the HTTP header remains open.

`ccharset` - the character set to use.

Generates

Content-type: <`ccontent_type`>; charset=<`ccharset`>

Example

```
owa_util.mime_header('text/plain', false, 'ISO-8859-4')
```

generates:

Content-type: text/plain; charset=ISO-8859-4\n

owa_util.print_cgi_env procedure

Syntax

```
owa_util.print_cgi_env;
```

Purpose

This procedure generates all the CGI environment variables and their values made available by the PL/SQL cartridge to the stored procedure.

Parameters

None.

Generates

A list in the following format:
cgi_env_var_name = value\n

owa_util.redirect_url procedure

Syntax

```
owa_util.redirect_url(
    curl          in      varchar2
    bclose_header in      boolean     DEFAULT TRUE);
```

Purpose

This procedure specifies that the application server is to visit the specified URL. The URL may specify either a web page to return or a program to execute.

This procedure must come before any htp or htf procedure or function call.

Parameters

curl - the URL to visit.

bclose_header - whether or not to close the HTTP header. If TRUE, two newlines are sent, which closes the HTTP header. Otherwise, one newline is sent, and the HTTP header is still open.

Generates

Location: <curl>\n\n

owa_util.showpage procedure

Syntax

```
owa_util.showpage;
```

Purpose

This procedure prints out the HTML output of a procedure in SQL*Plus, SQL*DBA, or Oracle Server Manager. The procedure must use the `htp` or `htf` packages to generate the HTML page, and this procedure must be issued after the procedure has been called and before any other HTP or HTF subprograms are directly or indirectly called. This method is useful for generating pages filled with static data.

Note that this procedure uses `dbms_output` and is limited to 255 characters per line and an overall buffer size of 1,000,000 bytes.

Parameters

None.

Generates

The output of `htp` procedure is displayed in SQL*Plus, SQL*DBA, or Oracle Server Manager. For example:

```
SQL> set serveroutput on
SQL> spool gretzky.html
SQL> execute hockey.pass( "Gretzky" )
SQL> execute owa_util.showpage
SQL> exit
```

This would generate an HTML page that could be accessed from Web browsers.

owa_util.showsource procedure

Syntax

```
owa_util.showsource (cname in varchar2);
```

Purpose

This procedure prints the source of the specified procedure, function, or package. If a procedure or function which belongs to a package is specified, then the entire package is displayed.

Parameters

cname - name of the procedure or function.

Generates

The source code of the specified function, procedure, or package.

owa_util.signature procedure

Syntax

```
owa_util.signature;  
owa_util.signature (cname in varchar2);
```

Purpose

This procedure generates an HTML line followed by a signature line on the HTML document. If a parameter is specified, the procedure also generates a hypertext link to view the PL/SQL source for that procedure. The link calls the [owa_util.show-source procedure](#).

Parameters

cname - the function or procedure whose source you want to show.

Generates

Without a parameter, the procedure generates a line that looks like the following:

This page was produced by the **PL/SQL Agent** on August 9, 1995 09:30.

With a parameter, the procedure generates a signature line in the HTML document that might look like the following:

This page was produced by the **PL/SQL Agent** on 6/14/95 09:30

[View PL/SQL Source](#)

owa_util.status_line procedure

Syntax

```
owa_util.status_line(
    nstatus      in      integer,
    creason     in      varchar2 DEFAULT NULL,
    bclose_header in      boolean DEFAULT TRUE);
```

Purpose

This procedure sends a standard HTTP status code to the client. This procedure must come before any `htp.print` or `htp.prn` calls so that the status code is returned as part of the header, rather than as “content data”.

Parameters

`nstatus` - the status code.

`creason` - the string for the status code.

`bclose_header` - whether or not to close the HTTP header. If TRUE, two newlines are sent, which closes the HTTP header. Otherwise, one newline is sent, and the HTTP header is still open.

Generates

Status: <`nstatus`> <`creason`>\n\n

owa_util.tablePrint function

Syntax

```
owa_util.tablePrint(
    ctable      in      varchar2
    cattributes in      varchar2    DEFAULT NULL
    ntable_type in      integer     DEFAULT HTML_TABLE
    ccolumns    in      varchar2    DEFAULT '*'
    cclauses    in      varchar2    DEFAULT NULL
    ccol_aliases in      varchar2    DEFAULT NULL
    nrow_min    in      number     DEFAULT 0
    nrow_max    in      number     DEFAULT NULL) return boolean;
```

Purpose

This function generates either preformatted tables or HTML tables (depending on the capabilities of the user's browser) from database tables. Note that RAW columns are supported, but LONG RAW columns are not. References to LONG RAW columns will print the result 'Not Printable'. In this function, *cattributes* is the second, rather than the last, parameter.

Parameters

ctable - the database table.

cattributes - other attributes to be included as-is in the tag.

ntable_type - how to generate the table. Specify "HTML_TABLE" to generate the table using <TABLE> tags or "PRE_TABLE" to generate the table using the <PRE> tags.

ccolumns - a comma-delimited list of columns from *ctable* to include in the generated table.

cclauses - WHERE or ORDER BY clauses, which let you specify which rows to retrieve from the database table, and how to order them.

ccol_aliases - a comma-delimited list of headings for the generated table.

nrow_min - the first row, of those retrieved, to display.

nrow_max - the last row, of those retrieved, to display.

Generates

A preformatted or HTML table.

Returns TRUE if there are more rows beyond the nrow_max requested, FALSE otherwise.

Example

For browsers that do not support HTML tables, create the following procedure:

```
create or replace procedure showemps is
  ignore_more boolean;
begin
  ignore_more := owa_util.tablePrint('emp', 'BORDER', OWA_UTIL.PRE_TABLE);
end;
```

and requesting a URL like this example: <http://myhost:8080/ows-bin/hr/plsql/showemps> returns to the client:

<PRE>

| EMPNO | ENAME | JOB | MGR | HIREDATE | SAL | COMM | DEPTNO |
|-------|--------|-----------|------|-----------|------|------|--------|
| 7369 | SMITH | CLERK | 7902 | 17-DEC-80 | 800 | | 20 |
| 7499 | ALLEN | SALESMAN | 7698 | 20-FEB-81 | 1600 | 300 | 30 |
| 7521 | WARD | SALESMAN | 7698 | 22-FEB-81 | 1250 | 500 | 30 |
| 7566 | JONES | MANAGER | 7839 | 02-APR-81 | 2975 | | 20 |
| 7654 | MARTIN | SALESMAN | 7698 | 28-SEP-81 | 1250 | 1400 | 30 |
| 7698 | BLAKE | MANAGER | 7839 | 01-MAY-81 | 2850 | | 30 |
| 7782 | CLARK | MANAGER | 7839 | 09-JUN-81 | 2450 | | 10 |
| 7788 | SCOTT | ANALYST | 7566 | 09-DEC-82 | 3000 | | 20 |
| 7839 | KING | PRESIDENT | | 17-NOV-81 | 5000 | | 10 |
| 7844 | TURNER | SALESMAN | 7698 | 08-SEP-81 | 1500 | 0 | 30 |
| 7876 | ADAMS | CLERK | 7788 | 12-JAN-83 | 1100 | | 20 |
| 7900 | JAMES | CLERK | 7698 | 03-DEC-81 | 950 | | 30 |
| 7902 | FORD | ANALYST | 7566 | 03-DEC-81 | 3000 | | 20 |
| 7934 | MILLER | CLERK | 7782 | 23-JAN-82 | 1300 | | 10 |

</PRE>

To view just the employees in department 10, and only their employee ids, names, and salaries, create the following procedure:

```
create or replace procedure showemps_10 is
  ignore_more boolean;
begin
  ignore_more := owa_util.tablePrint
    ('EMP', 'BORDER', OWA_UTIL.PRE_TABLE,
     'empno, ename, sal',
     'where deptno=10 order by empno',
     'Employee Number, Name, Salary');
end;
```

A request for a URL like **http://myhost:8080/ows-bin/hr/plsql/showemps_10** would return the following to the client:

```
<PRE>
-----
| Employee Number |Name| Salary |
-----
| 7782| CLARK| 2450 |
| 7839| KING | 5000 |
| 7934| MILLER | 1300 |
-----
</PRE>
```

For browsers that support HTML tables, to view the department table in an HTML table, create the following procedure:

```
create or replace procedure showdept is
  ignore_more boolean;
begin
  ignore_more := owa_util.tablePrint('dept', 'BORDER');
end;
```

A request for a URL like **http://myhost:8080/ows-bin/hr/plsql/showdept** would return the following to the client:

```
<TABLE BORDER>
<TR>
<TH>DEPTNO</TH>
<TH>DNAME</TH>
<TH>LOC</TH>
</TR>
```

```
<TR>
<TD ALIGN="LEFT">10</TD>
<TD ALIGN="LEFT">ACCOUNTING</TD>
<TD ALIGN="LEFT">NEW YORK</TD>
</TR>
<TR>
<TD ALIGN="LEFT">20</TD>
<TD ALIGN="LEFT">RESEARCH</TD>
<TD ALIGN="LEFT">DALLAS</TD>
</TR>
<TR>
<TD ALIGN="LEFT">30</TD>
<TD ALIGN="LEFT">SALES</TD>
<TD ALIGN="LEFT">CHICAGO</TD>
</TR>
<TR>
<TD ALIGN="LEFT">40</TD>
<TD ALIGN="LEFT">OPERATIONS</TD>
<TD ALIGN="LEFT">BOSTON</TD>
</TR>
</TABLE>
```

which a Web browser can format to look like this::

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO

owa_util.todate function

Syntax

```
owa_util.todate(p_dateArray in dataType) return date;
```

Purpose

This function converts the [owa_util.dateType data type](#) to the standard Oracle database DATE type.

Parameters

p_dateArray - the value to convert.

Generates

A standard DATE.

owa_util.who_called_me procedure

Syntax

```
owa_util.who_called_me(
    owner          out      varchar2
    name           out      varchar2
    lineno         out      number
    caller_t       out      varchar2);
```

Purpose

This procedure returns information (in the form of output parameters) about the PL/SQL code unit that invoked it.

Parameters

owner - the owner of the program unit.

name - the name of the program unit. This is the name of the package, if the calling program unit is wrapped in a package, and the name of the procedure or function if the calling program unit is a stand-alone procedure or function. If the calling program unit is part of an anonymous block, this is NULL.

lineno - the line number within the program unit where the call was made.

caller_t - the type of program unit that made the call. The possibilities are: package body, anonymous block, procedure, and function. Procedure and function are used only for stand-alone procedures and functions.

Generates

Not applicable.

`owa_util.who_called_me` procedure

Index

A

address tag, 1-6
anchor, 1-7
applet tags, 1-2, 1-9
area tag, 1-11

B

base tag, 1-12
basefront tag, 1-13
bgsound tag, 1-14
big tag, 1-15
blockquote tag
 closed, 1-16
 open, 1-16
body tag, 1-17
bold tag, 1-18

C

center tag
 tags
 center, 1-19
centerclose tag, 1-20
centeropen tag, 1-20
CGI environment variables, 9-11, 9-20
character formatting tags (PL/SQL cartridge), 1-4
checkbox form, 1-37
cite tag, 1-21
code tag, 1-22
comments in HTML, 1-23

D

data type
 owa_content.bigstring_arr, 2-3
 owa_content.number_arr, 2-25
 owa_content.string_arr, 2-36
dfn tag, 1-24
dirlistclose tag, 1-25
dirlistopen tag, 1-25
div tag, 1-26
dlistDef tag, 1-28
dlistTerm tag, 1-29
download_file, 1-30

E

emphasis tag, 1-32
environment variables
 retrieving in the PL/SQL Cartridge, 9-11
escape_sc, 1-33
escape_url, 1-34

F

fontClose tag, 1-35
fontOpen tag, 1-35
fonts
 big, 1-15
 small, 1-86
form tags, 1-2
formCheckbox, 1-37
formClose tag, 1-38
formHidden, 1-39
formImage, 1-40

formOpen tag, 1-38
formPassword, 1-41
formRadio, 1-42
formReset, 1-43
formSelectClose, 1-44
formSelectOpen, 1-44
formSelectOption, 1-46
formSubmit, 1-47
formText, 1-48
formTextarea, 1-49
formTextarea2, 1-49
frame tag, 1-53
frame tags, 1-5
framesetClose, 1-54

G

get_download_files_list, 1-31

H

head tag, 1-55
heading tag, 1-56
HTML comment, 1-23
htmlClose tag, 1-57
htmlOpen tag, 1-57
htp.address, 1-6
htp.anchor, 1-7
htp.anchor2, 1-7
htp.appletclose, 1-9
htp.appletpopen, 1-9
htp.area, 1-11
htp.base, 1-12
htp.basefont, 1-13
htp.bgsound, 1-14
htp.big, 1-15
htp.blockquoteClose, 1-16
htp.blockquoteOpen, 1-16
htp.bodyClose, 1-17
htp.bold, 1-18
htp.br, 1-73
htp.center, 1-19
htp.centerClose, 1-20
htp.centerOpen, 1-20
htp.cite, 1-21

htp.code, 1-22
htp.comment, 1-23
htp.dfn, 1-24
htp.dirlistClose, 1-25
htp.dirlistOpen, 1-25
htp.div, 1-26
htp.dlistClose, 1-27
htp.dlistDef, 1-28
htp.dlistOpen, 1-27
htp.dlistTerm, 1-29
htp.em, 1-31
htp.emphasis, 1-31
htp.fontClose, 1-35
htp.fontOpen, 1-35
htp.formCheckbox, 1-37
htp.formClose, 1-38
htp.formHidden, 1-39
htp.formImage, 1-40
htp.formOpen, 1-38
htp.formPassword, 1-41
htp.formRadio, 1-42
htp.formReset, 1-43
htp.formSelectClose, 1-44
htp.formSelectOpen, 1-44
htp.formSelectOption, 1-46
htp.formSubmit, 1-47
htp.formText, 1-48
htp.formTextarea, 1-49
htp.formTextareaClose, 1-51
htp.frame, 1-53
htp.framesetClose, 1-54
htp.framesetOpen, 1-54
htp.headClose, 1-55
htp.header, 1-56
htp.headOpen, 1-55
htp.hr, 1-63
htp.htmlClose, 1-57
htp.htmlOpen, 1-57
htp.img, 1-58
htp.img2, 1-58
htp.isindex, 1-60
htp.italic, 1-61
htp.kbd, 1-62
htp.keyboard, 1-62
htp.line, 1-63

htp.linkRel, 1-64
htp.linkRev, 1-65
htp.listHeader, 1-66
htp.listingClose, 1-67
htp.listingOpen, 1-67
htp.listItem, 1-68
htp.mailClose, 1-70
htp.mailto, 1-69
htp.mapOpen, 1-70
htp.menuListClose, 1-71
htp.menuListOpen, 1-71
htp.meta, 1-72
htp.nl, 1-73
htp.nobr, 1-74
htp.noframesClose, 1-75
htp.noframesOpen, 1-75
htp.olListClose, 1-76
htp.olListOpen, 1-76
htp.para, 1-77
htp.paragraph, 1-77
htp.param, 1-78
htp.plaintext, 1-79
htp.preClose, 1-80
htp.preOpen, 1-80
htp.print, 1-81
htp.prints, 1-82
htp.prn, 1-81
htp.ps, 1-82
htp.s, 1-83
htp.sample, 1-84
htp.script, 1-85
htp.small, 1-86
htp.strike, 1-87
htp.strong, 1-88
htp.style, 1-89
htp.sub, 1-90
htp.sup, 1-91
htp.tableCaption, 1-92
htp.tableClose, 1-95
htp.tableData, 1-93
htp.tableHeader, 1-94
htp.tableOpen, 1-95
htp.tableRowClose, 1-96
htp.tableRowOpen, 1-96
htp.teletype, 1-97

htp.textareaOpen, 1-51
htp.textareaOpen2, 1-51
htp.title, 1-98
htp.ulistClose, 1-99
htp.ulistOpen, 1-99
htp.underline, 1-100
htp.variable, 1-101
htp.wbr, 1-102

I

img tag, 1-58
IP address
 retrieving in the PL/SQL cartridge, 7-3
isindex, 1-60
italic tag, 1-61

J

Java applets
 referencing, 1-9

K

keyboard tag, 1-62

L

line tag, 1-63
link tag, 1-64
list item tag, 1-68
list tags, 1-2
listHeader tag, 1-66
listingClose tag, 1-67
listingOpen tag, 1-67

M

mailto tag, 1-69
map tags, 1-70
menu list tag, 1-71
meta tag, 1-72

N

new line tag, 1-73

no break tag, 1-74
noframes tags, 1-75

O

ordered list tag
tags
 olist, 1-76
owa_content package, 2-1
owa_content.bigstring data type, 2-3
owa_content.document_exists, 2-7
owa_content.get_attribute, 2-8
owa_content.get_attributes, 2-11
owa_content.get_author, 2-12
owa_content.get_char_set, 2-13
owa_content.get_content_type, 2-14
owa_content.get_description, 2-15
owa_content.get_encoding, 2-16
owa_content.get_expiration, 2-17
owa_content.get_language, 2-18
owa_content.get_length, 2-19
owa_content.get_title, 2-20
owa_content.list_attributes, 2-21
owa_content.list_documents, 2-22
owa_content.list_system_attributes, 2-23
owa_content.list_user_attributes, 2-24
owa_content.number_arr data type, 2-25
owa_content.rename_document, 2-26
owa_content.set_attribute, 2-27
owa_content.set_author, 2-28
owa_content.set_char_set, 2-29
owa_content.set_content_type, 2-30
owa_content.set_description, 2-31
owa_content.set_encoding, 2-32
owa_content.set_expiration, 2-33
owa_content.set_language, 2-34
owa_content.set_title, 2-35
owa_content.string_arr datatype, 2-36
owa_cookie.cookie data type, 3-2
owa_cookie.get function, 3-3
owa_cookie.get_all procedure, 3-4
owa_cookie.remove procedure, 3-5
owa_cookie.send procedure, 3-6
owa_cookie_vc_arr data type, 3-2
owa_image.get_x function, 4-4

owa_image.get_y function, 4-5
owa_image.NULL_POINT package variable, 4-2
owa_image.point data type, 4-3
owa_opt_lock.checksum function, 5-3
owa_opt_lock.get_rowid function, 5-4
owa_opt_lock.store_values procedure, 5-5
owa_opt_lock.vcArray data type, 5-2
owa_opt_lock.verify_values function, 5-6
owa_pattern.amatch function, 6-4
owa_pattern.change function and procedure, 6-6
owa_pattern.getpat procedure, 6-8
owa_pattern.match function, 6-9
owa_pattern.pattern data type, 6-11
owa_sec.get_client_hostname function, 7-2
owa_sec.get_client_ip function, 7-3
owa_sec.get_password function, 7-4
owa_sec.get_user_id function, 7-5
owa_sec.set_authorization procedure, 7-6
owa_sec.set_protection_realm procedure, 7-8
owa_text.add2multi procedure, 8-2
owa_text.multi_line data type, 8-3
owa_text.new_row_list, 8-4
owa_text.print_multi procedure, 8-5
owa_text.print_row_list procedure, 8-6
owa_text.row_list data type, 8-7
owa_text.stream2multi procedure, 8-8
owa_text_vc_arr data type, 8-9
owa_util.bind_variables function, 9-3
owa_util.calendarprint procedure, 9-4
owa_util.cellsprint procedure, 9-6
owa_util.choose_date procedure, 9-9
owa_util.dateType data type, 9-10
owa_util.get_cgi_env function, 9-11
owa_util.get_owa_service_path function, 9-12
owa_util.get_procedure function, 9-13
owa_util.http_header_close procedure, 9-14
owa_util.ident_arr data type, 9-15
owa_util.ip_address data type, 9-16
owa_util.listprint procedure, 9-17
owa_util.mime_header procedure, 9-19
owa_util.print_cgi_env procedure, 9-20
owa_util.redirect_url procedure, 9-21
owa_util.showpage procedure, 9-22
owa_util.showsource procedure, 9-23
owa_util.signature procedure, 9-24

owa_util.status_line procedure, 9-25
owa_util.tablePrint function, 9-26
owa_util.todate function, 9-30
owa_util.who_called_me procedure, 9-31

regular expressions, 6-2
strong tag, 1-88
style tag, 1-89
subscript tag, 1-90
superscript tag, 1-91

P

paragraph formatting tags (PL/SQL cartridge), 1-3
paragraph tag, 1-77
parameter tag, 1-78
PL/SQL cartridge
 applet tags, 1-2
 character formatting tags, 1-4
 form tags, 1-2
 frame tags, 1-5
 list tags, 1-2
 paragraph formatting tags, 1-3
 subprograms summary, 1-1
 table tags, 1-3
PL/SQL Web Toolkit
 htf package, 1-1
 htp package, 1-1
 owa_cookie package, 3-1
 owa_image package, 4-1
 owa_opt_lock package, 5-1
 owa_pattern package, 6-1
 owa_sec package, 7-1
 owa_text package, 8-1
 owa_util package, 9-1
plaintext tag, 1-79
preformatted text tag, 1-80
print tag, 1-81

T

table tags (PL/SQL cartridge), 1-3
tableCaption tag, 1-92
tableClose tag, 1-95
tableData tag, 1-93
tableHeader tag, 1-94
tableOpen tag, 1-95
tableRowClose tag, 1-96
tags
 address, 1-6
 applet, 1-9
 area, 1-11
 base, 1-12
 basefront, 1-13
 bgsound, 1-14
 big, 1-15
 blockquote
 closed, 1-16
 open, 1-16
 body, 1-17
 bold, 1-18
 centerclose, 1-20
 centeropen, 1-20
 cite, 1-21
 code, 1-22
 dfn, 1-24
 dirlistclose, 1-25
 dirlistopen, 1-25
 div, 1-26
 dlistDef, 1-28
 dlistTerm, 1-29
 emphasis, 1-32
 fontClose, 1-35
 fontOpen, 1-35
 formClose, 1-38
 formOpen, 1-38
 frame, 1-53
 head, 1-55

R

regular expressions, 6-2
revision tag, 1-65

S

sample tag, 1-84
script tag, 1-85
small font tag, 1-86
strike tag, 1-87
strings

header, 1-56
htmlClose, 1-57
htmlOpen, 1-57
img, 1-58
italic, 1-61
keyboard, 1-62
line, 1-63
link, 1-64
listHeader, 1-66
listitem, 1-68
mapClose, 1-70
mapOpen, 1-70
menulistClose, 1-71
menulistOpen, 1-71
meta, 1-72
nl, 1-73
nobr, 1-74
noframes, 1-75
paragraph, 1-77
parameter, 1-78
plaintext, 1-79
pre, 1-80
print, 1-81
revision, 1-65
sample, 1-84
script, 1-85
small, 1-86
strike, 1-87
strong, 1-88
style, 1-89
sub, 1-90
sup, 1-91
tableCaption, 1-92
tableData, 1-93
tableHeader, 1-94
teletype, 1-97
title, 1-98
underline, 1-100
unordered list, 1-99
variable, 1-101
wbr, 1-102
teletype tag, 1-97
title tag, 1-98

U

underline tag, 1-100
unordered list tag, 1-99

V

variable tag, 1-101
vc_arr data type, 3-2

W

wbr tag, 1-102